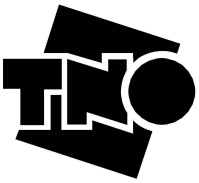
Adjustable Autonomy

Controling Influences on Decision Making



The research reported in this thesis was partly conducted at TNO Defense, Security and Safety, Den Haag, The Netherlands.

The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project (http://www.icis.decis.nl/), supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

SIKS Dissertation Series No. 2009-20

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Printed by Gildeprint drukkerijen B.V., Enschede

ISBN 978-90-5986-322-4

Adjustable Autonomy

Controling Influences on Decision Making

Verstelbare Autonomie

Beheersen van Invloeden op Besluitvorming

(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Universiteit Utrecht
op gezag van de rector magnificus, prof.dr. J.C. Stoof,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen
op maandag 6 juli 2009 des middags te 4.15 uur

door

Bob van der Vecht

geboren op 29 januari 1981, te Apeldoorn

Promotor: Prof.dr. J.-J.Ch. Meyer
Co-promotor: Dr. F.P.M. Dignum

Contents

Chapter 1

Introduction

This research is motivated by a perspective on automation of distributed systems. As systems are becoming more capable of performing complex tasks, a number of new applications can be thought of where different actors collaborate to reach joint goals. Collaboration can be achieved in several ways and coordination of action always plays an important role. However, as the actors are autonomous entities, it is not obvious to achieve coordinated behavior.

This study deals with single-agent reasoning and multi-agent organizations. Both subjects are getting a lot of attention by researchers, but there is still enough to be explored. The combination of the two areas is interesting, because where current multi-agent organizational research mainly focuses on the organizational perspective without single-agent issues, single-agent research stresses individual reasoning, but leaves out organizational aspects. We believe that both areas are strongly related and influence each other. Therefore, a combined approach can lead to new understandings.

We are interested in dynamic coordination between autonomous actors. Our approach to achieve dynamic coordination in multi-agent systems is to take an individual perspective on those organizations. We study the relation between local autonomy and global coordination. According to our definition, an autonomous agent determines by itself to what extend it is being influenced by external factors. We define adjustable autonomy as the process to dynamically allowing or ignoring influences. As a result agents achieve coordination by allowing influences on their decision making. Therewith, adjustable autonomy plays a key role in achieving dynamic coordination.

In this chapter, we describe the background of this research project. We describe our research goal and our approach.

1.1 Project Background

The research presented here is part of the Interactive Collaborative Information Systems (ICIS) project [, ICIS]. The goal of the ICIS project is to study techniques and methods that help to build information systems with intelligent behavior. The ICIS project uses a focus on one particular application domain, namely crisis management.

Crisis management systems deal with complex, dynamic environments, where quick response plays an important role. As this PhD project is part of the ICIS project, we will link our research and results to applications in the crisis management domain. These results however, can be used in other application areas as well. Application domains are characterized by complex, dynamic and chaotic environments, where groups of actors work together.

When thinking of future systems in such environments, several capabilities are required to guarantee well functioning. The system should be able to cope with unexpected events and with uncertainty of information. It should adapt to the circumstances and act in real time. Furthermore, one of the system tasks in the crisis management domain is to give support to human actors. For example, process information and search for relevance, meaning and importance.

1.1.1 ICIS Scenario

A typical ICIS scenario can be constructed around an accident in a traffic tunnel. For example, a truck catches fire while driving through a busy tunnel. Several crisis management organizations become involved: fire brigade, medical services, police, and other organizations as well. There are other people involved as well: possible victims, spectators and remaining traffic.

All crisis organizations share a common goal, namely to manage the crisis. At the same time they have their own tasks: the fire brigade fights the fire, while the medical services takes care of the victims and the police re-routes the traffic. Continuously, the organizations need to coordinate their actions with each other. For example, as the medical services need a designated location for the first aid to the victims, the police needs to keep the public away from this area. If the firemen expect explosion danger, they need to inform the other organizations to evacuate.

Information sharing is important to achieve coordination and technology can play a role here. First of all, the tunnel can be equipped with a sensor network. We think about sensors that are no longer passive data collectors, but can perform actions and communicate with each other. An information management system shows up, that can combine data, derive information and distribute it within the organization.

The human beings in the crisis management organizations can be supported by technology. For example, autonomous robots can search the crisis area on victims, or collect data at places that are inaccessible for humans. Also, decision

support systems can be useful. Assume that the crisis workers are equipped with PDA's that are directly connected to the information management system. The information system has situation awareness and has knowledge about what information is relevant for the crisis workers. It can provide the crisis workers with personalized information directly.

One can think of several ways in which the technology can support the humans, as well in which it can take actions by itself to fight the crisis. The technology in such crisis management organization plays a active role. It is part of the organization. The ICIS project studies the techniques and methods that enable the development of such systems.

1.1.2 Collaborative Decision Making

Within the ICIS project, the study presented here was part of the Collaborative Decision Making (CDM) cluster. Decision making is an everyday activity. Decision-making involves gathering, interpreting and assessing information, formulating and judging alternatives and choosing a course of action that will fulfill a certain objective. Decision-making is not solely an individual activity, but also occurs at group level. Within the ICIS project, CDM is considered as a dynamic decision-making process, in which all participants share the responsibility to achieve a common objective.

Examples of CDM existing in human organizations are team discussions that lead to group decisions. As result of technological developments we foresee future systems where organizations of artificial actors coordinate and negotiate with each other. Examples are sensor networks where sensors are embedded in entities that can perform actions and exchange information, or gaming and simulation applications with advanced virtual characters. CDM will also occur in hybrid organizations. In human-machine interaction the machine will act more as a team member instead of a user tool. Such hybrid organizations, in which agents and humans cooperate in a dynamic manner, require new perspectives on organizational design for both the human and the artificial organization [Klein et al., 2004]. The main aim of the CDM cluster is to study the technical implications on system design when we are dealing with a human-agent organization.

1.2 Adaptive Support Systems

The original title of this research was *Adaptive Support Systems*. Adaptive support systems are support systems that are capable of altering their role or function in an organization according to changing operational demands. The role of the system in organizations will change: from static information providers and task performers to pro-active support systems that can adapt to changing demands coming from the environment or the organization.

As systems are becoming more capable of performing complex tasks, we expect more cooperation between systems to reach joint goals. Cooperation can be achieved in several ways, but coordination of action always plays an important role. Adaptive support systems coordinate their actions in a dynamic manner. Adaptivity, here, refers to the ability of systems to modify its structure or behavior to obtain a better fit when the circumstances change.

In order to describe the aims of this research more clearly, we address the following questions first: What are support systems? What is adaptivity?

1.2.1 What is a Support System?

We consider each system which provides support to the user to be a support system. This can be support on demand or pro-active support. However, in this research we do not want to stick to the traditional user-system view. We want to focus on interactive systems consisting of multiple agents, human beings as well as artificial agents. The definition we have found for support system, which represents our view is: *A support system is a network of facilities and people who interact for mutual assistance.*

There are various routes to achieve new levels of support, ranging from a straightforward focus on automation of tasks, to approaches in which the level of support alters adaptively [Neef, 2006]. The former requires research decisions on which tasks are eligible for automation, and how such automated tasks can be fitted in existing organizations. The latter, generally called adaptive automation, involves the design of support systems that are capable of altering their role or function in an organization according to changing operational demands. We believe that support systems are steadily evolving into flexible assistants and are becoming more a part of the organizations than a set of supporting tools.

Research on adaptive automation is increasing nowadays [Neef, 2006] and gives the support system a far more active role in decision making and information processing. According to the researchers, the employment of pro-active, smart technology, such as intelligent agents, will affect the way hybrid teams perform their tasks. Intelligent agents will not only support humans adaptively, but will also be capable to perform complex tasks independently, perhaps even to the extent of replacing humans as team members altogether.

1.2.2 What is Adaptivity?

The term *adaptive* does not have a clear definition. There are many forms of adaptivity. Adaptiveness can be seen as a system feature when it reacts to changes in the environment. Within this context adaptiveness is the opposite of static. However, it depends on the perspective of the observer whether something is considered adaptive or not. In this research, we make a distinction between an adaptive system and adaptive behavior of a system.

An example of an adaptive system is a system that learns from experience [Neal, 1996]. The system updates internal response parameters in reaction to the outcome of an action. In contrary, a static system stays internally the same over time and it does not have the ability to change. When a system reacts to an event in the environment, it adapts its behavior to the environmental change. We can consider the observed behavior to be adaptive. From that view, a robot driving forward, which suddenly sees a wall in front of him, and as a reaction turns right, is showing adaptive behavior.

From this perspective, a static system can show adaptive behavior. The robot in the example itself can be a completely static system. Also, an adaptive system can show static behavior, for example a system that continuously adapts its internal structure but always shows a constant output. It might very well be the developer's aim that the internal adaptation results in a constant output.

Whether a behavior is adaptive or not, depends on the perspective from which you observe the system and the environment. An example is a car with cruise control, driving with constant speed uphill and downhill. If you look only at the car's speed from the outside, you see static behavior, but if you also consider the engine as part of the car, you notice adaptive behavior. What if the same car is driving with a constant engine power, which makes it move slower uphill than downhill? If you look now only at the car from the outside, you see adaptive behavior, but if you consider the engine as well, you see static behavior.

It depends on the perspective of the observer whether something is considered adaptive or not. In this project, we discuss adaptive support systems. An adaptive support system together with its user can be seen as a human-agent team that exhibits adaptive capabilities. The team should be able to alter role and task allocations (internal adaptivity) and course of action (external adaptivity). Because the team includes artificial actors, this implies that it should be possible to have artificial actors switch roles with human actors, and vice versa. The aim of this research is to facilitate such adaptive features.

1.2.3 Mixed-Initiative Systems

When combining support systems with adaptive behavior, we come to *mixed-initiative systems* [Bradshaw et al., 2004]. In mixed-initiative systems several types of collaboration occur in a dynamic manner. Mixed-initiative means that the initiative for actions of the system comes from multiple actors. Mixed-initiative systems can consist of human and artificial actors. Human beings can collaborate with artificial actors, or several types of artificial actors can collaborate with each other. Dynamic task allocation in such systems is studied by Klein, et al. [Klein et al., 2004]. The researchers list ten challenges that an artificial actor should meet in order to become a team player in a human-agent team setting.

In this research, we adopt some system requirements of mixed-initiative systems and we try to meet these requirements using an agent-based approach. A

property of mixed-initiative systems is that the system makes use of different types of collaboration among the actors. Figure 1.1 shows some of the concepts that actors in mixed-initiative systems should be aware of in order to handle different coordination types dynamically. In this research we focus on the issue of autonomy, although neither autonomy, nor any of the other requirements can be studied in isolation. The various concepts interact and influence each other and add to the complexity of this research topic.

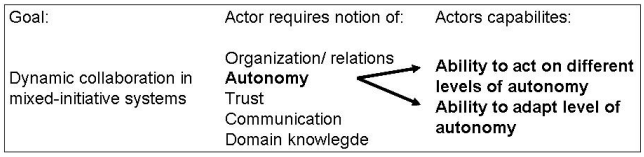


Figure 1.1: *Important issues concerning dynamic collaboration in mixed-initiative systems*

Intuitively, the concept of autonomy is related to the ability of taking initiative. In human cooperation, people readily change their level of dependence with respect to others. This can be regarded as adaptation of their autonomy level. We expect artificial actors in mixed-initiative systems to possess adjustable autonomy as well. However, this ability is currently lacking. At this point, we choose for an agent-based approach to study the concept of adaptive support systems. Next, we describe the ideas of agent-based systems and discuss the concept of autonomy in more detail.

1.3 Agent-based Systems

Agents are commonly used in artificial intelligence as a design paradigm for systems. However, the properties that are used to describe the features of an agent-based system are hard to capture. For example, agent-based systems are said to demonstrate *intelligent behavior*. Other properties are easier to understand, such as the *distributed* character of an agent-based system.

When developing systems that meet the requirements of adaptive support systems, an agent-based approach seems to be a logical choice. Concepts that are used in the agent community are recurring themes when dealing with the design of adaptive support systems.

It appears hard give an exact definition of an agent. Agent-based systems are often presented in contrast to object-oriented systems, e.g. [Jennings, 2000]. The definition from Wooldridge and Jennings [Wooldridge and Jennings, 1995] is among the most cited ones: *An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives.* In their paper,

the researchers attempt to define what an agent is, and therewith, the following features keep recurring:

- agents are autonomous
- agents show social ability
- agents are reactive
- agents are pro-active

Being autonomous is understood as having the ability to operate without human intervention. In this aspect agents show contrast with systems or applications that are controlled by people, who continuously monitor and modify the system's behavior. We believe that the autonomy feature puts demands on the internal design of agents.

Showing social ability implies that agents are able to communicate and coordinate with other agents of with human beings. Communication is the process of exchanging information between a sender and a receiver. In cooperative settings communication leads to tuned activities between different parties.

Reactivity implies that the agent is able to sense the environment and act according to its observations. Fully reactive systems directly link observations to actions. They act reflexive to their perceptions. Examples of such systems are Braitenberg creatures [Braitenberg, 1984], very simple robots that relate sensor input directly to movement actuators.

The pro-activity feature implies that the agent acts goal-directed. The agent chooses its actions based on internal motivations and tries to achieve desired world states. This is often seen as a requirement for rationality [Russell and Norvig, 2003]. Goal-directed behavior makes sure that an agent acts in a consistent way with respect to a longer-term goal. It is opposite to reactive behavior, where immediate response to observations is dealt with.

There is no exact definition of agent and there are several other approaches to agent-based systems. However, for the purpose of this research the definition of Wooldridge and Jennings [Wooldridge and Jennings, 1995] seem to fit to our needs. All agent features described above contribute to intelligent behavior of an agent. Several other features are mentioned, such as situatedness, flexibility, the ability to learn. Many of them return in our research goals to design adaptive support systems. Adaptive support systems are expected to provide different levels of support, and to adapt their level to the circumstances. The system takes the initiative to perform actions. Therefore, the system has some degree of autonomy. Autonomy is a key concept in agent research as well; in Chapter 2 of this thesis we will discuss several studies on agent autonomy.

1.4 Autonomy

We have identified *autonomy* as a key concept of adaptive support systems, as well as of artificial agents. In this research we try to get a grip on this concept. In different contexts autonomy can means different things. We present a perspective based on philosophical grounds. The term *autonomy* comes from the Greek words Auto-Nomos, which means *one who gives oneself his/her own law.* It can be explained as the ability of self-government. It refers to the capacity of an individual to make a its own decisions without external intervention.

The word *autonomy* has several usages in philosophical contexts. One of the leading philosophical theories of autonomy was developed by Kant in his work *Groundwork of the Metaphysic of Morals* [Kant, 1785], where he sees autonomy as self-governance. An autonomous person decides on its own actions based on internal desires and considerations. Kant applied this concept to create a definition of person hood. As a consequence, he uses autonomy as the basis for determining responsibility for one's actions. Kant uses the autonomy property of humans to define a universal theory of moral law. Therefore, autonomy refers to a person's capacity for self-determination in moral choices.

This view on autonomy, where it is strongly related to responsibility with respect to actions, should be distinguished from autonomy as a state that one could want to achieve. The ideal state to be autonomous implies authenticity; to be free from any form of manipulation or influence by others [Dworkin, 1988]. This ideal state is unreachable for human beings in practice, as it is generally accepted that one's development is influenced by social circumstances. It does not match with social capabilities. In an artificial system, however, the ideal state to be autonomous can be achieved by design choices.

In this research we discuss autonomy of artificial systems in a social, interactive context. Therefore, we adopt the notion of autonomy as self-governance. This choice has consequences on the design of the system. On the one hand, it should be able to show social behavior and take other actors in consideration. On the other hand, it should be responsible for its actions. The balance between those requirements suggests that there are degrees of some kind in autonomy, and that the system can adapt its autonomy level. In Chapter 2 we discuss several approaches of autonomy in agent research.

We take autonomy in agent decision making as its primary focus of research. We use adjustable autonomy as the driver force for other multi-agent issues, such as coordination, collaborative problem solving and human-agent teamwork.

1.5 Research Approach

In this research we study dynamic coordination in hybrid, dynamic organizations. The organizations consist of heterogeneous actors, who need to coordinate their actions in order to reach the organizational goals. The type of collaboration

between the actors is dynamic. In this research we take a system perspective to develop a model for an artificial actor in such system. We adopt the agent paradigm as basis for the actors.

1.5.1 Research Goal

The goal of this research is to develop an agent model that describes actors that are capable to function in a dynamic organization. The agent should be autonomous, such that it can be held responsible for its actions, it should be able to work together with others following different types of collaboration, and it should be able to adapt the collaboration type to achieve dynamic coordination.

1.5.2 Research Questions

In order to reach the research goal, we start with an investigation of other studies on agent autonomy and coordination. We take a position in how the autonomy requirement should be represented in an agent model, and we describe the required components and interaction between them. Furthermore, we investigate the effect of possible coordination mechanisms on agent autonomy. **Chapter 2** discusses the following questions:

- What is autonomy in the context of agents? What is adjustable autonomy? What are the necessary components in an agent reasoning-model in order to meet the autonomy requirement? What is the effect of the chosen coordination mechanisms on the autonomy of individual agents?

The answer to these questions leads to a definition of autonomy and an abstract model for agent reasoning, such that the autonomy requirement is met and coordination can be achieved. We define *being autonomous* as *having control over external influences on the decision-making process.* This means that the agent determines by itself to what extend its decisions are being influenced by external factors. The required components of a reasoning model include a component for influence control and a component for decision making. Coordination can be achieved by explicitly allowing external influences on the decision making. Adjustable autonomy is a key issue in achieving dynamic coordination.

In chapter 3, we try to validate the proposed reasoning model. We give a possible implementation of the reasoning model. Furthermore, we analyze different types of inter-agent influences and we investigate how group coordination can be achieved via influence control by individual agents. We set up an experiment where a group of agents has to fulfill a coordinated task, to test whether different forms of coordination can be achieved. The questions that are answered in **Chapter 3** are:

- How can we implement the components in the reasoning model? What type of external influences does an agent experience? In what ways can an agent

process external events? Can we show that coordination can be achieved based on influence control?

As result in Chapter 3 we propose an implementation of the component for influences control based on reasoning rules for event processing. We propose an implementation of the decision-making component using a Belief-Desire-Intention (BDI) approach. We analyze what attitudes an agent can take with respect to the environment and to other agents. The coordination experiment demonstrates that coordination can be achieved via local influence control of the individual agents. Furthermore, it shows that dynamic coordination can be achieved by allowing the agents to adapt their attitude with respect to external influences.

This raises the question based on which heuristics the agent can process external events. We propose three basic heuristics, where the agent processes external events based on information relevance, trust and organizational knowledge. These heuristics are discussed in Chapters 4 and 5. Furthermore, it becomes clear that for dynamic coordination the agents need to control in a dynamic way how they are influenced. This shows the need for a metareasoning process about event-processing, which is subject of Chapter 6.

Chapter 4 addresses the information-relevance heuristic to process external events. We discuss the following questions:

- What are the benefits of processing external events based on information relevance? How can an agent determine relevance of information?

We analyze the use of information relevance in several scenarios and we evaluate the benefits in a computational model. Furthermore, we present an algorithm with which BDI agents can determine relevance of information with respect to their goals. We present a working implementation of an agent using information relevance to process incoming events.

In **Chapter 5** we discuss event processing based on organizational knowledge. We try to connect local autonomy and global coordination by showing that coordination can be achieved by influencing each other. The research questions in this chapter are:

- How should a coordination mechanism be specified, such that it respects local autonomy? How can an autonomous agent adopt organizational rules? Can organizational rules be translated to event-processing rules of an agent? Can our approach handle organizational dynamics?

We argue that in order to guarantee local autonomy, coordination models should be defined separately from the internal specification of the agent. The OperA [Dignum, 2004] model for organizational specification fits to our needs. We present a way to translate organizational rules from OperA to event-processing rules of agents. We show that the event-processing rules can be adapted easily, as they are defined separately from the decision-making process. Therefore, the

agents are able to adopt changes in organizational rules, and thus are able to handle reorganization. Furthermore, the agents remain autonomous, and therewith they are able to initiate bottom-up dynamics in the organization.

Finally, we study the processing of controling external influences. We have described heuristics based on which the agent can process external events. However, the agent should decide when to use which heuristic. The reasoning model consists of a component for event processing and a component for decision making. The modular approach allows for feedback from the decision-making component to the event-processing component. In **Chapter 6** we develop a metareasoning model to facilitate this feedback.

- Can we develop a model for metareasoning about event processing? What are required components in a metareasoning model? What issues should be considered when implementing? Can we regulate adaptivity such that it improves performance for both individual and organization?

Based on a general model for metareasoning we propose an extension to our reasoning model, which allows reasoning about event-processing at a metalevel based on the current performance of the agent. We present an implementation of the model and we describe how to predict the adaptive behavior of an agent based on the implementation of the metareasoning.

In **Chapter 7** we discuss and conclude the results presented in the other chapters. Furthermore, we describe several application areas where the results of the research can be used. We give examples for applications in support systems, organizational modeling and gaming and simulation.

1.6 Embedded Articles

Although this thesis has been written as a book, it is based on earlier publications. The perspective on autonomy as having control over external influences was introduced in Vecht et al. [Vecht et al., 2007] and it evolved into theory of Chapters 2 and 3. Different perspectives on agent autonomy are discussed in Vecht et al. [Vecht et al., 2008b], which has been extended to Chapter 2, and part of the paper recurs Chapter 3 as well. Work about an algorithm to determine information relevance was the basis for Chapter 4 [Vecht et al., 2008a]. The relation between influence control in autonomous agents and behavioral rules according to organizational specifications has been addressed earlier as well [Vecht et al., 2009] and has been extended to Chapter 5.

Chapter 2

Autonomy, Adjustable Autonomy and Coordination

This chapter discusses the concepts *agent autonomy* and *adjustable autonomy*. In the previous chapter we argued that an agent-based approach is a valid choice when studying adaptive support systems. We presented definitions of autonomy from philosophy. Here, we discuss autonomy in the context of agent research. We compare different approaches of autonomy and adjustable autonomy. Motivated by other work, we present our definition and our approach. The definition for autonomy that we adopt, states that being autonomous means to have control over the reasoning process, and, as a consequence, to have control over external influences on the decision-making process.

Further in this chapter, we discuss the relation between autonomy and coordination. Coordination can be described as a balance between local autonomy and global control. As we define autonomy as having control over external influences, coordination can be achieved by allowing certain influences on the decision-making process.

2.1 Autonomy in Agent Research

The topics agent autonomy and adjustable autonomy have been subject of many studies. However, there is no common definition of autonomy. As a result, the approaches taken to deal with the concept are quite distinct. In this section we discuss several ways autonomy is used in related work on agent research. We investigate whether adjustable autonomy exists in the approaches and what adjustability means in the different perspectives that are taken.

2.1.1 Autonomy as Defining Feature

Autonomy is often mentioned in definitions of agents. Wooldridge and Jennings [Jennings, 2000; Wooldridge and Jennings, 1995] use autonomy as a defining feature of agents, together with other features such as *situatedness*, *pro-activeness*, *reactive* and *social*. Autonomy in their view means that the agent has control over both its internal state and its behavior. In this definition, autonomy is a feature an entity should meet in order to be an agent. Autonomy has no gradual properties, thus adjustable autonomy does not exist.

The requirement to be autonomous is that the agent *has control over its behavior and internal state*. This does not mean than it is free from any external influences, but the agent should control them. The researchers do not specify in what way the autonomy requirement has implications for the reasoning process of the agent.

In their paper, d'Inverno and Luck pose a theoretical view on autonomy [d'Inverno and Luck, 1998]. Autonomy is a feature of agents, without dimension, measure or degree. An autonomous agent is independent with respect to user and designer. The view on autonomy of d'Inverno and Luck as a defining feature is comparable to the philosophical view on autonomy as a state of authenticity, which implies no manipulation or influence by others. An autonomous agent will be solipsistic. This is a strong view, which is not practical in a social context, where, for example, actors need to cooperate to achieve a common goal. This definition of autonomy is stronger than the definition of Wooldridge and Jennings [Jennings, 2000; Wooldridge and Jennings, 1995].

2.1.2 Autonomy as Abstraction Levels of Decision Making

Castelfranchi and Falcone, [Castelfranchi, 1995; Falcone and Castelfranchi, 2001], present a view on autonomy with gradual properties. Autonomy can be seen as abstraction levels of the decision-making process. Autonomy is considered in the context of relations between agents. Considering a hierarchical relation, a delegate has a certain level of autonomy with respect to his master. The level of autonomy determines which aspects of the decision-making process are done by the delegate and which are done by the master. Within the decision-making process several abstraction levels can be distinguished; *action execution*, *planning* and *goal determination*. Consequently, the following levels of autonomy exist:

- *executive autonomy*: the agent is not allowed to decide anything but the execution of delegated task
- *planning autonomy*: the agent is allowed to plan (partially); the delegated task is not fully specified
- *goal autonomy*: the agent is allowed to find its own goals

Verhagen [Verhagen, 2000] has added norm autonomy as an extra level of abstraction. A norm-autonomous agent determines its own norms, and the norms may vary over time. This implies that the agent can reason about organizational norms, and it is able to violate them. Norm autonomy requires that agents have the capability of reasoning about the organizational structures in the system.

Adjustable autonomy is the process of switching between the autonomy levels. The researchers do not present an implementation of how autonomy could be modeled within an agent. However, if we relate this view on autonomy to a reasoning model, the different abstraction levels refer to the internal concepts used in the reasoning process (goals, plans, actions). The level of autonomy specifies whether the concepts are controlled by the agent, or by some external actor.

2.1.3 Social Autonomy

Falcone and Castelfranchi [Falcone and Castelfranchi, 2001] call their approach to autonomy *social autonomy*, as it always concerns relations between agents. A social-autonomous agent can decide whether to accept or reject a delegated task. This is an internal process. The autonomy of the agent with respect to its superior increases along various dimensions. The more open the delegation (the less specified the task is), the more autonomous an agent is from its superior with respect to that task. In the same way, the more control actions are given up by the superior, or the more decisions are delegated by the superior, the more autonomous an agent is.

In this case, adjustable autonomy is the process of changing the dependency with respect to another agent. [Falcone and Castelfranchi, 2001] lists some reasons why someone could want to change his own autonomy:

- The actor comes to believe that it is not able to do all the task (e.g. lack of self-confidence)
- The actor foresees problems because of unexpected (external) events
- The actor believes he can do better with more autonomy
- The actor wants to 'surprise' another agent by doing something extra

The first two are reasons for decreasing one's autonomy level, the last two are reasons to increase the autonomy. Falcone and Castelfranchi state that the autonomy in decision-making is bilaterally adjustable. This means that an agent's autonomy is adjustable by the agent himself and by other agents as well. Reasons to adjust the autonomy level consist of expectations of an agent to fulfill its task successfully or not.

Carabelea and Boissier [Carabelea et al., 2004] describe the relational property of autonomy as well: an agent is autonomous to some degree with respect to another agent or user or norm. They distinguish external and internal perspectives

on autonomy: the external perspective concerns what is observable, the internal perspective concerns what happens inside the agent and how the agent makes its decisions.

The researchers aim to develop a decision-making module that will allow the creator of an agent to explicitly specify how the agent makes the decisions when it has autonomy and how these decisions are made for it when it has no autonomy, both for social and norm autonomy. However, this has not been fully specified.

d'Inverno and Luck [d'Inverno and Luck, 1998] present another relational view on autonomy, where autonomy is related to independence. Here, a non-autonomous agent has a fixed level of dependence with respect to others. It can even mean that it is fully independent. To be autonomous, however, the agent should be able to switch between levels of dependency. They define autonomous agents to be agents that generate their own goals from motivations. If you consider goals of an agent, it is the self-generation of goals that implies autonomy.

2.1.4 Autonomy as Ability and Permission of Actions

Bradshaw et al., [Bradshaw et al., 2004, 2005] are looking at agent autonomy on the level of actions; action performance and action selection. According to them autonomy consists of two dimensions; the ability to perform actions alone and the permission of choosing actions alone.

The spectrum of actions in the general case consists of the following aspects. Potential actions are all theoretical actions in a world. Possible actions are all potential actions regarding a certain situation. And performable actions are possible actions of a certain actor, and thus taking the actor in account as well. If an agent can perform all possible actions, it is fully autonomous according to the first dimension of the definition; it has the ability to perform all possible actions.

A social context defines whether actions are permitted or obliged. If all potential actions are permitted, an agent is fully autonomous according to the second dimension; it can always choose every performable action.

Adjustable autonomy is equivalent to resizing the performable and permitted actions of the agent. In a social context it means reducing or increasing the permissions and obligations. Looking at the individual agent, changing the agent's capabilities or changing the environment leads to other values for performable and possible actions.

In their theory, the autonomy of an agent is adjusted externally, so the agents have no self-adjustable autonomy. Adjustment of agent autonomy is specifying how agent behavior should be constrained. Their work on adjustable autonomy mainly concerns human-agent interaction and policies are specified to make actions permitted, obliged, forbidden. Those policies are implemented as hard-coded norms. Adjustable autonomy is equivalent to adjusting capabilities and permissions. Their focus is on user-agent relations. In their theory, the autonomy of an agent is adjusted externally (by a user). Therefore, the agents themselves have no notion of autonomy and they do not adjust their autonomy levels by themselves.

2.1.5 Autonomy as Action Deliberation

A different view on autonomy is presented in [Dastani et al., 2004b]. Dastani, et al. argue that the deliberation cycle of an agent determines the autonomy of an agent. Autonomy levels can be viewed at as an agent's commitment to its own decisions. For example, a deliberation cycle that makes an agent to commit to a goal until it has been fulfilled is different from a cycle that makes an agent to reconsider its goals every cycle based on new observations. In the first case the agent is more autonomous considering its goals than the second one, as it does not allow any external factors to influence its goals.

They propose a meta-language to describe the deliberation cycle of an agent. The functions used in the deliberation cycle as well as their actual implementation are relevant for agent autonomy. Levels of autonomy can be constructed changing the deliberation cycle. The authors do not give a model for adjustable autonomy. In this approach, levels of autonomy are determined by the deliberation cycle, and therefore by the way decisions are made.

2.1.6 Autonomy as Influence on Group Decisions

We already mentioned social autonomy, which describes agent autonomy in relation to another agent. Other studies look at autonomy in a multi-agent context, and focus on the process of group decision-making.

Barber and Martin, [Barber et al., 2000; Martin and Barber, 2006], are looking at the decision-making process of a group of agents. An agent's level of autonomy with respect to a task is measured as its share in the group decision-making process. They define the autonomy of a single agent with respect to the goal as: *An agent's degree of autonomy, with respect to some goal that it uses its capabilities to pursue, is the degree to which the decision-making process, used to determine how that goal should be pursued, is free from intervention by any other agent.*

They introduce a numerical model to calculate an agent's autonomy. Consider a group of agents for a decision-making process. Autonomy is described as a tuple $< G, D, C >$, where G is the focus of the autonomy assignment (a set of goals/tasks), D identifies the decision-makers and their relative strength in the decision making process, and C declares the authority constraint of the decision-making process, i.e. the agents that will possibly carry out the decisions.

For example: Agent A_1, A_2 and A_3 are thirsty and want to buy milk. They know that they can get milk if one of them goes to the shop to buy it. $G = buymilkA_1, buymilkA_2, buymilkA_3$. A_1 and A_2 will negotiate about who will perform the task, and therefore $D = A_1, A_2$. Agent A_3 will commit to the outcome, so $C = A_1, A_2, A_3$. Say that A_1 and A_2 have same negotiation strength. Then the autonomy measure of A_1 and A_2 with respect to G will be 0.5 and of A_3 will be 0.

The way to measure autonomy as presented here is done from an organizational

view. Thus, take into account a number of agents, a set of goals and an authority constraint, and one can measure autonomy per agent with respect to the decision-making process of the whole group.

In their context adjustable autonomy concerns different decision-making strategies for a group of agents. They present an Adaptive Decision-Making Framework [Martin and Barber, 2006], in which agents propose strategies to the group, and therewith change their own autonomy level. This way, adjustable autonomy becomes a group process, because other agents can accept or reject proposed decision-making strategies. The researchers do not specify how an agent can determine the preferred decision-making strategies in different situations. In the experiments they conducted they provided the agents with knowledge about the best strategy for each situation.

2.1.7 Autonomy as Decision-Making Control

Another work on adjustable autonomy in a multi-agent context is [Scerri et al., 2004] by Scerri et al. They use the term adjustable autonomy for the process in which a decision maker transfers the control of the decision-making process to another agent (or human). The agents have pre-planned transfer-of-control strategies that contain actions to transfer the decision-making control or change the coordination constraints (e.g. ask for more time to make the decision). Time-outs are used as reasons to transfer decision-making control.

The researchers do not give a definition of the concept of autonomy, but it is related to decision-making control with respect to a certain goal. A coordination mechanism that runs independently from the agent decision-making, handles the transfer-of-control process.

The researchers do not use autonomy as a gradual property of the decision-making process of an agent. Their reasoning mechanism for adjustable autonomy can only be used when there are more agents that have the capability to make the decision. The mechanism should make sure the optimal decision maker is selected. Coordination consists of passing on tasks or asking for time extension.

2.2 Comparing the Perspectives on Autonomy

As we have seen so far in the discussion of related work there are no standard definitions of agent autonomy and adjustable autonomy. We have described some approaches of autonomy in agent research. In this section we make a comparison of the different approaches by identifying differences and similarities. We generalize to two approaches of autonomy: *influences on decision making*, and *freedom of choice*. We investigate the relation of the approaches with respect to a reasoning model. Furthermore, we compare the adjustability component and the agent's awareness about its autonomy level.

2.2.1 External Influences on Decision-Making

The general idea of autonomy that keeps recurring is that the agent autonomy is related to the external influences on the decision-making process. However, the type of influences and the way they are handled is quite different for the distinct approaches.

When taking autonomy as a defining feature, there is the requirement that the agent controls its behavior and its internal state [Jennings, 2000; Wooldridge and Jennings, 1995], or even that the decision-making is free from external influences [d'Inverno and Luck, 1998]. It is assumed that an agent fulfills this property.

The perspective on autonomy as a relational issue [Carabelea et al., 2004; Falcone and Castelfranchi, 2001], consider the influence of one agent on the decisions of another agent in a social context. The focus is on task delegation as a specific type of influence. The influence of organizational norms on an agent's decisions is described as norm autonomy [Verhagen, 2000]. Bradshaw et al. [Bradshaw et al., 2004] take autonomy as the ability and permission of actions, and then an external actor influences the agent by modifying the options it can choose from.

The definition of autonomy used by Martin and Barber [Martin and Barber, 2006] explicitly uses external influences on a decision to determine the autonomy of an agent. They apply the notion to group decisions, in contrast to others who are concerned with single-agent decisions.

2.2.2 Freedom of Choice

Related to influences on decision making is the idea that autonomy means *freedom of choice*. A fully autonomous is agent is free to do whatever it wants. However, the environment can restrict the freedom of choice of the agent, and therewith, limits its autonomy. This notion of autonomy occurs in Bradshaw et al. [Bradshaw et al., 2004]. They define autonomy as the ability and permission of actions.

Depending on the reasoning model of an agent, rules for coordination can restrict the freedom of choice of an agent via obligations and norms. The ability to reason about delegated tasks and norms determines whether the autonomy is directly restricted by the coordination rules. The abstraction level of decision making [Falcone and Castelfranchi, 2001; Verhagen, 2000], plays a role here. If the agent explicitly chooses to which norms it conforms, it is not just the norm that limits the autonomy of the agent, because the agent has the choice to obey or not.

2.2.3 Relation to the Reasoning Process

Another comparison can be made on the effect of an autonomy definition on a reasoning model. The definition of autonomy that is chosen can have implications for the decision-making process, both on single-agent and multi-agent decision-

making. We can distinguish three aspects of reasoning that are relevant: the way the local decision-making process is influenced by others, the way the decisions are actually made, and the way it relates to multi-agent decision-making. Here we discuss the implications on the reasoning model for single and multi-agent decision making.

Single-Agent Reasoning

We notice that some of the approaches point to different aspects of a single-agent reasoning process. Different aspects of decision-making are taken into account, such as abstraction levels of decision making, [Falcone and Castelfranchi, 2001; Verhagen, 2000], actions [Bradshaw et al., 2004, 2005] and action deliberation [Dastani et al., 2004b]. Figure 2.1 shows in a graphical representation of a reasoning-process that different approaches point to different locations in the model.

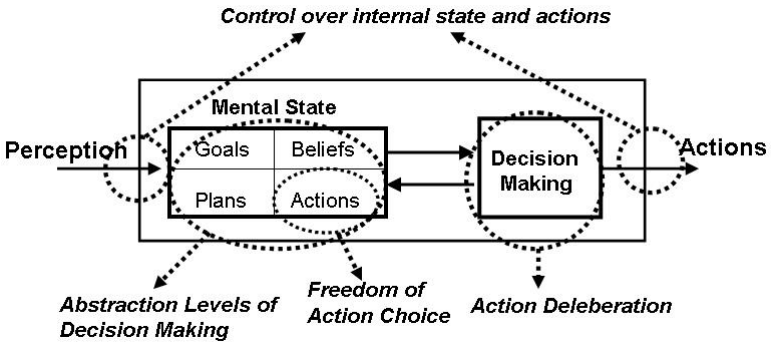


Figure 2.1: *Different approaches of autonomy point to different locations in a reasoning model*

The picture shows that several aspects of the reasoning model can be taken into account when talking about autonomy. It also shows that some approaches of autonomy do not conflict. For example, restricting the ability and permission of actions does not say anything about *how* the actions are chosen in a deliberation cycle. Some approaches can exist next to each other or might even reinforce each other.

Multi-Agent Reasoning

Besides the effect of autonomy on the reasoning process of a single agent, we have seen that other approaches of autonomy consider several multi-agent aspects of autonomy. Falcone and Castelfranchi discuss autonomy in the context of one-on-one relations [Falcone and Castelfranchi, 2001], where the level autonomy is

always considered with respect to someone else. Verhagen deals with abstract organizational aspects such as norms [Verhagen, 2000]. Group decision-making is discussed as well [Martin and Barber, 2006; Scerri et al., 2004]. Martin and Barber relate one's autonomy level to one's influence on a group decision; a different group decision-making style results in a different autonomy level. For Scerri et al., autonomy reflects to the control over a (group) decision, instead of to the decision-making style itself. Figure 2.2 visualizes the place where the approaches point to in a multi-agent context.

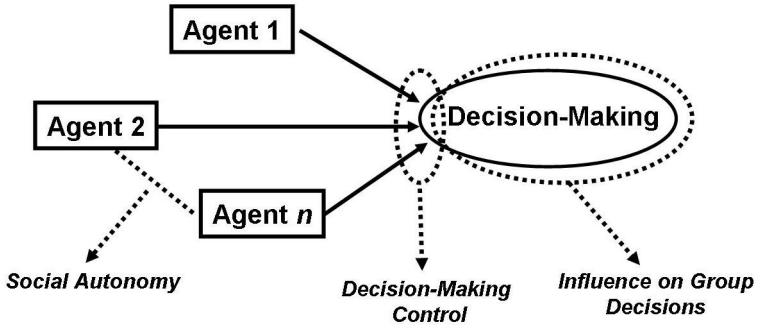


Figure 2.2: *Different approaches of autonomy point to different locations in a reasoning model*

2.2.4 Adjustability

Another aspect that can be compared in the different approaches is *adjustability*. When talking about autonomy as a defining feature [d'Inverno and Luck, 1998; Jennings, 2000; Wooldridge and Jennings, 1995], no levels are distinguished and there is no adjustable aspect of autonomy. However, other views on autonomy do have an adjustable component; levels of abstraction [Falcone and Castelfranchi, 2001; Verhagen, 2000], ability and permission of actions [Bradshaw et al., 2004, 2005] and adjustable deliberation cycles [Dastani et al., 2004b].In the work focusing on multi-agent decision-making, adjustable autonomy concerns varying decision-making style [Martin and Barber, 2006] or transfer of the decision-making control [Scerri et al., 2004]. Table 2.1 gives an overview of the different approaches

of agent autonomy and the corresponding autonomy levels and their adjustability feature.

Use of Autonomy	Levels of Autonomy	Adjustable Autonomy (AA)
Control over internal state and behaviour	No	No
Abstraction Levels of Decision Making	Execution, plan, and goal autonomy	Switch between abstraction levels
Social Autonomy	Levels of delegation	Switch between delegation levels
Ability and Permission of actions	Two dimensions: ability and permission	Vary within dimensions
Action Deliberation	Frequency of reconsidering choices	Vary deliberation cycle
Influence on Group Decision-Making	Local Autonomy, Consensus, Command Driven	Switch between decision-making strategies
Decision-Making (DM) control	No	Transfer DM-control to others

Table 2.1: *Different approaches of autonomy and adjustable autonomy*

The control over autonomy adjustment can be inside or outside the agent. In their work, Bradshaw et al. [Bradshaw et al., 2004, 2005] give the adjustability control to an external actor. However, self-adjustable autonomy is very well possible according to their autonomy definition. For example, an agent can influence its abilities by manipulating the environment. Falcone and Castelfranchi [Falcone and Castelfranchi, 2001] discuss social autonomy, and according to them, the autonomy level is adjustable by the agent itself as well as by an external actor. In the case of varying the deliberation cycle [Dastani et al., 2004b], it is very well possible to give the control over the autonomy to the agent itself.

2.2.5 Awareness of Autonomy

Another notable issue is the agent's *awareness* with respect to its autonomy. In the philosophical context, discussed in Chapter 1, autonomy is taken as basis for responsibility. Therefore, the assumption is that an agent makes rational decisions, and is aware of its choices with respect to moral laws. The way Wooldridge and Jennings [Jennings, 2000; Wooldridge and Jennings, 1995] use autonomy in agent definitions matches with this view. They state that an autonomous agent should have control over both its internal state and its behavior. The use of the word *control* here, can be interpreted in different ways. Control can be taken as

having the power to manipulate; it can be achieved by predefined, hard-coded rules. In a more cognitive approach, having control implies a motivated decision. Therefore, the agent should be aware of the influences it experiences and it should be aware of its reasoning process.

When we look at the models of autonomy presented in related work, this latter interpretation is not always the guaranteed. For example, if an agent's autonomy level is controlled outside of the agent as in [Bradshaw et al., 2004, 2005], the agent does not need to have any notion of its autonomy level. Also, in a hierarchical relation, where the master can adjust the autonomy of the delegee [Falcone and Castelfranchi, 2001], the delegee need not be aware of it.

The model of Martin and Barber [Martin and Barber, 2006], the agent has the possibility to calculate its autonomy with respect to some decision. They consider group decisions instead of single-agent decisions. Therefore, it has a different link to responsibility.

2.3 Autonomy as Influence Control

In the previous sections, we have discussed related work. We described several approaches to agent autonomy and adjustable autonomy and we tried to compare some issues of the different approaches. In this section, we present a view on autonomy that is related to some of the presented work, but has a new perspective. We focus on the aspect of *having control over external influences on the decision-making process*. We motivate this perspective based on the discussion of the related work in Section 2.2.

2.3.1 A Definition of Autonomy

In Chapter 1, we motivate to use agent-based systems in our aim to develop adaptive support systems. The philosophical view on autonomy, that is serves as bases for responsibility, was an important reason to do so. The way Wooldridge and Jennings [Jennings, 2000; Wooldridge and Jennings, 1995] describe the autonomy requirement matches with the philosophical view. They state that an autonomous agent should have control over both its internal state and its behavior. Having control over the behavior implies that the agent has a local decision-making process. Having control over the internal state implies that the agent controls how it is being influenced by other agents and by the environment.

Wooldridge and Jennings do not specify the implications of the autonomy requirement for an agent reasoning model. In sections 2.1 and 2.2 we discussed several other approaches that try to develop a reasoning model that meets the autonomy requirement. The approaches discussed in Section 2.2 have in common that autonomy is about how external factors influence the decision-making process. However, the focus in general is on actions and on action selection.

We think that the aspect of having control over the internal state has had too little attention in other work on agent autonomy. We want to argue that beliefs influence the behavior of an agent in an indirect way. This issue is emphasized as well by Castelfranchi [Castelfranchi, 1995]. If an agent does not control its internal state, it is sensitive for manipulation by external factors, and therefore, its autonomy is not guaranteed.

The same holds for human beings. If a person is subject to a lot of propaganda, that person might start believing specific things unconsciously. In that case the person has lost (part of the) control over his internal state and might not be held responsible for his actions. The mental state and the behavior of this person is manipulated by other people influencing his beliefs.

This notion also applies to agents. Imagine an agent that observes its environment and determines its actions autonomously based on beliefs about the environment. If an external party knows how the agent reacts on observations and it controls the environment, the external party can control the behavior of the agent. In a social context, an agent that communicates with other agents in a multi-agent system is for part of its knowledge dependent on others, and might be sensitive for manipulation via information sharing.

We assume that agents have a local decision-making process with which they determine their actions. In order to guarantee the autonomy of the agent, it should judge the external influences it experiences instead of passively allowing them. Controling the internal state means that the agent chooses which concepts it uses for its decision-making. Therewith, the agent controls to what extend it is being influenced by external factors. This is our focus to facilitate autonomy in an agent reasoning-model. We define being autonomous as *having control over external influences on the decision-making process.*

Definition 2.1 *Being* ***autonomous*** *means having control over external influences on the decision-making process.*

The definition implies that an agent consciously determines how it is being influenced by other agents and by the environment. By stating that the agent controls the influences, we demand that the agent is aware of them. If an agent makes decisions based on beliefs, goals and plans, it should control how these concepts are influenced by external factors.

An agent that takes a static attitude towards the environment might actively judge external influences. However, it does not show any adaptivity with respect to the situation. We believe that the process of controling influences should be explicitly part of agent reasoning. Therewith, the agent shows adjustable autonomy.

2.3.2 A Definition of Adjustable Autonomy

Work on adjustable autonomy has in common that it focuses on action selection [Bradshaw et al., 2004; Falcone and Castelfranchi, 2001; Martin and Barber, 2006; Scerri et al., 2004]. Our focus will be on all types of influence by other agents. Our hypothesis is that an autonomous agent should control all external influences on its decision-making process. These external influences can be goals, plans and norms, but also beliefs. Adjustable autonomy in our context means: *dynamically dealing with external influences on the decision-making process based on internal motivations.* Our definition of adjustable autonomy implies that it focuses on *self-adjustable autonomy* and that the agent is aware of its autonomy level.

Definition 2.2 ***Adjustable Autonomy****: Dynamically dealing with external influences on the decision-making process based on internal motivations.*

Adjustable autonomy is the actual process of controling external influences. It gives the agent the opportunity to change its openness towards other agents and towards the environment. Therewith, adjustable autonomy plays an important role in coordination of a group of agents.

In the other studies little is said about reasons to adjust one's autonomy. Martin and Barber [Martin and Barber, 2006] provide the agents with the best decision-making style for any situation. However, this is only possible in an environment with finite possible states that are all known in advance. Scerri et al. [Scerri et al., 2004] use time limits as triggers to adjust autonomy. Falcone and Castelfranchi [Falcone and Castelfranchi, 2001] suggest performance expectations to adjust one's autonomy. We need to work out a practical model for self-adjustable autonomy. For example, we can let an agent use heuristics to control its autonomy based on its internal state.

2.4 Autonomy and Reasoning

The definition of autonomy has implications of the reasoning model. We have defined autonomy as *having control over external influences on the decision-making process.* The choice of a reasoning model determines whether an agent meets the autonomy requirement. In this section, we introduce an abstract reasoning model that guarantees agent autonomy according to our definition. After that, we describe two other reasoning models, namely reactive reasoning and goal-directed reasoning, and discuss them in the context of the autonomy requirement.

2.4.1 An Abstract Reasoning Model

We stated that an autonomous agent should have control over external input it experiences. The agent should consciously allow external factors to influence its decision-making process. Our definition of autonomy focuses on the process of

controling external influences. Therefore, we want to include this process explicitly in a general reasoning model.

We distinguish a module for decision-making on actions, that gives the agent control over its behavior. Another module in the reasoning process gives the agent control over external influences. Figure 2.3 can be seen as the abstract representation of the reasoning model of an autonomous agent.

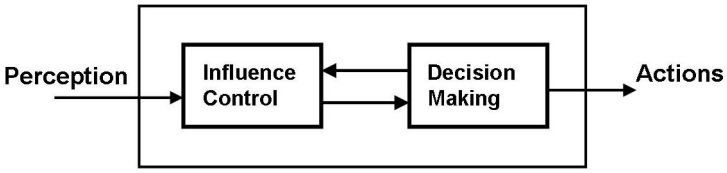


Figure 2.3: *General reasoning model for autonomous agents*

We have added an arrow from the decision-making component to the influence-control component in Figure 2.3. The connection implies that results from the decision-making component can have an effect on how external input is processed. Therewith, the agent can determine to what extend it is being influenced based on internal motivations. For example, it can choose to focus on specific information for a high-proirity task, or it can judge the degree of truth of messages based on it's confidence in the sender.

The picture presents an abstract model of reasoning. At this point, we do not say anything about the type of reasoning the agent does. If an agent makes decisions based on beliefs, goals and plans, it should be aware of how they are influenced by external factors and what the consequences are for its decisions. If the agent uses organizational norms to determine its actions, it should be aware how these norms influence its decisions. For our research purposes, we will use the abstract reasoning model as shown in Figure 2.3. There are several ways to implement the reasoning components. In the next sections we discuss two other reasoning models, namely reactive reasoning and goal-directed reasoning, and discuss them in the context of the autonomy requirement. In Chapter 3 we move toward an implementation choices for the reasoning model.

2.4.2 Reactive Reasoning

The first type of reasoning we want to discuss in the context of autonomy is reactive reasoning. We consider a simplistic type of reactive agents. The agents select their action directly based on the observations of the environment. Typically, a rule-based system is used for reactive reasoning, as shown in Figure 2.4.

Well known examples of reactive agents are Braitenberg creatures [Braitenberg, 1984]. Braitenberg defined robots that observed the world with only one sensor and controlled their wheels according to their observations. For external

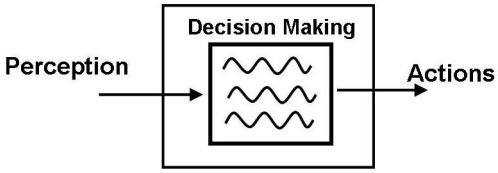


Figure 2.4: *Abstract model of reactive reasoning*

observers, the behavior of the robots seemed to contain some intelligence. For example, consider a robot with a light sensor and the reasoning rule that makes the robot drive away from the light. The behavior of the robot is to navigate away from the light at any time. By doing so, it shows behavior that resembles simple insect behavior.

In most applications, reactive agents are kept simple, and contain only few reasoning rules. A purely reactive agent has no explicit internal world model. They are used for large scale simulations where the individual agents show simple behavior, but focus is on the result of the interaction between the individuals. They are popular in research on emergence [Holland, 1998], ant-based systems and swarming [Eric Bonabeau and Theraulaz, 1999].

The advantage of reactive reasoning in agents is that it is fast. The agent reacts immediately to changes in the environment. With only a few reasoning rules an agent shows behavior that could make sense for an external observer. Drawback is that the agent's behavior is fully dependent on the environment. The reasoning rules are pre-determined and link observations directly to actions. All knowledge is included at design time and is captured in an implicit way by the reasoning rules. The consequence is that if situations are encountered that were not taken into account at design time, the agent might show unwanted behavior. Furthermore, the reasoning rules increase in number and in complexity when dealing with complex situations.

Let us look at autonomy in reactive agents. As the observations of the agent are linked directly to actions, we can say that the environment actually determines the agent's actions instead of the agent itself. The assumption then is that observations are passive; the agent perceives the world continuously with its sensors instead of actively selecting its observations. This assumption is met in practice as we look at reactive agent models. Therefore, we can conclude that a reactive agent is not autonomous in the definition as we use it.

It is possible to add a component for influence control to the reactive reasoning-model. The consequence would be that the agent no longer passively observes the environment, but it actively selects its observations. Furthermore, the observation processing needs to be controlled by the agent itself via feedback from the decision-making component. The new reasoning process is presented in Figure 2.5.

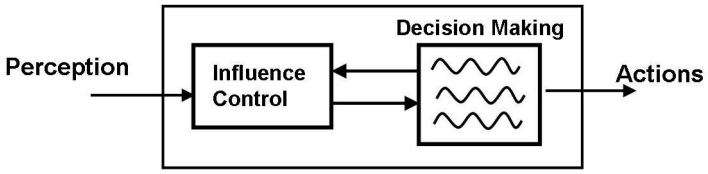


Figure 2.5: *Abstract model of reactive reasoning*

2.4.3 Goal-Directed Reasoning

Goal-directed reasoning models have been developed to have agents act in a proactive way. Goal-directed behavior makes sure that an agent acts in a consistent way with respect to a longer-term goal. Also, the goals are represented explicitly, and therefore may change over time. This is not possible in reactive agents, where goals implicitly exist in the reasoning rules.

A popular approach of goal-directed reasoning is the BDI-approach. The Belief-Desire-Intention (BDI) model for human reasoning was developed by Bratman [Bratman, 1987], and based on intentions theory of Dennett [Dennett, 1987]. The BDI-model can be used for programming intelligent agents [Rao and Georgeff, 1995]. After the implementation of an agent's initial beliefs, desires and intentions, it actually uses these concepts to observe the world and reason about actions. A BDI-agent is a bounded rational agent that has particular mental attitudes: Beliefs, Desires and Intentions. These attitudes form the agent's mental state. The agents reasons with its mental state to determine the desired actions. This is in contrast to reactive agents, that have no mental state, but react directly to observations.

Beliefs represent the informational state of the agent; its knowledge about the world including itself and other agents. Beliefs can also contain inference rules to derive new beliefs. Using the term belief rather than knowledge recognizes that what an agent believes may not necessarily be true in the world.

Desires represent the motivational states of the agent. They represent objectives or situations that the agent would like to accomplish. Usage of the term goals instead of desires adds the restriction that the set of goals must be consistent. For example, one should not have concurrent goals to go to a party and to stay at home, even though they could both be desirable. *Intentions* are considered to be desires or goals to which the agent has committed itself.

Plans are a fourth concept used in BDI reasoning. Plans are sequences of actions that an agent can perform to achieve its goals. Plans may include other plans as well as basic actions. In BDI models, the agent has often a plan library. The reasoning of the agent consists of selecting the right plans given its goals and beliefs. The process of creating plans is not captured by the standard BDI-model, but it could be included.

BDI-agents are goal-directed; they use goals explicitly to reason about the desired actions. This is in contrast to reactive agents, who link observations to actions directly. In BDI agents, observations influence the mental state. The agent determines its action by deliberation on the mental state. This is show in Figure 2.6.

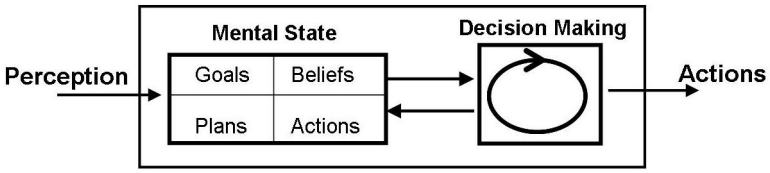


Figure 2.6: *Abstract model of BDI reasoning*

Looking at the autonomy of a BDI-agent, we can conclude that the agent determines its own actions, as it reasons with a mental state that is not fully determined by the environment. The agent has internal goals that ensure a level of consistency and determination in its choices; it does not simply react to its observations. The agent's behavior cannot be *controlled* directly by external factors, such as the environment or other agents. However, the behavior of the agent can be influenced via the internal state. The autonomy assumption requires that the agent has control over its internal state and the question is whether that holds or not. We would like to argue that the control over its internal state is not specified in the conceptual BDI reasoning model. Therefore, the BDI-model does not guarantee the autonomy requirement of agents. This is supported by practical implementations of BDI reasoning, discussed in Chapter 3, which show that it is common to adopt observations and messages automatically in the internal state.

In the same way we added a component for influence control in the reactive reasoning-model, we can add such a component in the BDI reasoning-model. Figure 2.7 shows the result in a graphical way. Here, the component for influence control determines the effect of external input on the mental state of the agent. Subsequently, the mental state is used to determine the actions of the agent. Controling external influences is an internal process, just as action selection. By explicitly adding influence control to the agent's reasoning-model, we facilitate the autonomy of the BDI-agent.

2.5 Autonomy and Coordination

Autonomy has a tight relation with coordination. An autonomous agent decides by itself upon its actions. Multi-agent systems consist of multiple autonomous actors that interact to reach a certain goal. In this section, we take a closer look

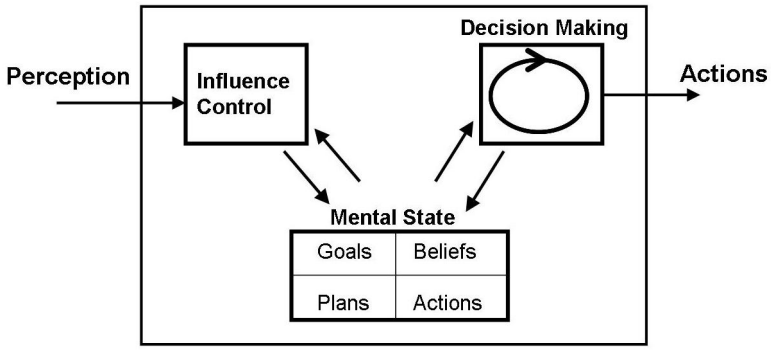


Figure 2.7: *Abstract model of BDI reasoning*

at multi-agent systems. We assume that agents are autonomous and we investigate the implications for multi-agent systems and particularly for coordination mechanisms. We look at coordination mechanisms in an abstract way, discussing emergent and controlled coordination, and we investigate higher-level coordination via organizational concepts.

2.5.1 Autonomy and Multi-Agent Systems

As stated in the previous section, autonomy is often used as a defining feature of agents. Here we investigate the implications of agent autonomy on multi-agent systems. Earlier in this chapter, we discussed the autonomy requirement saying that an autonomous agent has control over both its internal state and this behavior. This is compliant with the philosophical idea that autonomy means self-government. Therefore, autonomy consists of two parts:

- having control over the behavior
- having control over the internal state

Having control over the behavior implies that the agent has a local decision-making process. In the context of a multi-agent system, it guarantees the distributed feature of the system, as shown in Figure 2.8. The distributed feature of the system implies that there is not one single point of control. The system consists of several independent parts with local abilities to reason and to act.

Most approaches to agent reasoning we discussed in sections 2.1 and 2.2 ensure the feature that agents control their own behavior via local decision-making. However, we identified that *having control over the internal state* had little attention in related work. From that point of view we developed the definition

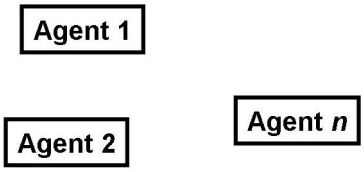


Figure 2.8: *Distributed control over behavior*

of autonomy that an autonomous agent should control external influences on its decision-making process.

This implies that the agent has local authority over how it is being influenced by other agents and by the environment. Together with the local decision-making, this feature enables self-governance and responsibility. Furthermore, the assumption that the agent controls influences on its internal state allows for social interaction and for coordination. As we look at coordination as gearing activities to one another, an agent accepts influence of other agents to its decision-making process. Autonomous agents achieve coordination by *allowing* external influences on local decisions. Therefore, a system consisting of multiple agents is not only distributed, but also coordinates and interacts, Figure 2.9.

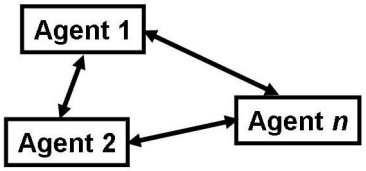


Figure 2.9: *Coordination and interaction by allowing influence on internal state*

These properties of a multi-agent system are based on the assumption that agents have control over their internal and over their behavior. However, this is only theory. In the next section, we discuss the practical implications of the autonomy assumption on coordination mechanisms.

2.5.2 Autonomy and Coordination Mechanisms

In Section 2.1, we already discussed some studies that focused on interaction in agents systems. Falcone and Castelfranchi wrote about social relations between agents [Falcone and Castelfranchi, 2001], Martin and Barber focused on group decision-making [Martin and Barber, 2006] and Scerri et al. achieve coordination via transfer of decision-making control [Scerri et al., 2004]. Here, we discuss more

general approaches of coordination. Not the actual decision-making process of the group is studied, but the behavior of the group. Several approaches can be taken, of which two extremes; *emergent* and *controlled* coordination.

One approach to reach coordinated group behavior is *emergent coordination* [Holland, 1998]. Actors perform their tasks independently and the interaction between many of them leads to coordinated behavior. This approach is often used for agent-based social simulations. One characteristic of emergent coordination is that the actors have no awareness of the goals of the organization they are part of. The actors make their own local decisions and are fully autonomous in the sense that they are free from any influence by other agents. The agents typically have a reactive reasoning model as discussed in Section 2.4.2.

Although the actors have no organizational awareness, the designer of such a system has. The coordination principles are specified implicitly within the local reasoning of all actors. The organization is relatively flexible within the single task for which it has been designed. However, in the extreme case, the agents are fully autonomous, and there is no point of control that can force the organization to change its behavior, if unexpected situations occur that cannot be solved by the local reasoning rules of the actors.

Where the fully emergent approach is one extreme type of coordination, the other extreme is fully *controlled coordination*. This is the case in a hierarchical organization, where there is a single point of control that determines the tasks all the others have to perform. The actors are autonomous in performing their task, but they do not make their own decisions. Therefore, the actors do not meet the autonomy requirement to control their own behavior.

A characteristic of such a centralistic approach is that the task division is made from a global perspective. Therefore an organization can adapt quickly to changes in the environment by sending out new orders to all actors. However, such an organization is sensitive to incomplete information. Wrong information at the global level can lead to wrong decisions. Furthermore, the organization is highly dependent on the decision maker at the top of the hierarchy and it misses the flexibility at the local level. Fully controlled coordination can be a good solution if there is always complete information about the situation. Task specifications and interaction protocols can be defined for all possible cases.

2.5.3 Autonomy and Organizational Models

In between the extreme types of emergent and controlled coordination, there are several ways to achieve coordination. Several organizational descriptions have been proposed. All organizations designed for a certain purpose require coordinated behavior of the participants. In this section we investigate the relation between autonomy of actors and coordination of behavior via organizational models. How do they ensure that the agent is aware of the organizational rules, without interfering with the agent's decision-making process? The different approaches for coordination have different consequences for the autonomy of the agents that

participate in the system.

One of the first was the Agent Group Role model [Ferber and Gutknecht, 1998], known as the AGR model, which introduced the concepts of roles, groups and interaction between roles. The model focuses on defining the structural aspects of the organization. The AGR model does not consider abstract behavior rules, such as norms. In the specification of the roles, the AGR model captures the individual agent behavior as it is desired by the organisation. All interaction between the roles is predefined and presented at low level. The agents in the AGR model are autonomous; they are outside of the organization. However, the organizational rules specify the desired behavior in terms of actions and goals. A normative approach to describe behavioral guidelines would leave the agents more space in choosing how they actually pursue the goals that belong to their role.

The OperA model [Dignum, 2004] proposes a more expressive way for defining organizations by introducing an organizational model, a social model and an interaction model. This approach explicitly distinguishes between the organizational model and the agents that act in it. Agents adopt roles in the organization via contracts that specify these rules, and therewith, they become aware of the organizational rules. The contracts describe the behavioral guidelines of a specific role. The internals of the agents are fully separated from the organizational specification. The only requirement is that the agents need to be able to understand the contracts. The agents are still fully autonomous in making decisions and in executing their actions.

Another approach is Moise+ [Hübner et al., 2002], which also separates the organizational model from the agent model. The organization is defined in terms of a structural, a functional and a deontic model. The structural model describes the roles, groups and relation between them. The functional model contains a task diagram for the common goal. The deontic model connects the other two models in a normative way. In order to operationalize the organization, middleware has been developed that checks whether actions of agents are allowed or not according to the governing organizational rules. The agents make decision locally, but the middleware can overrule the decisions of the agents. The organization becomes an active entity that has the possibility to interfere in the agents decisions. Therefore, the agents are not fully autonomous in controling their behavior.

Other organizational models, as introduced by Matson [Matson and DeLoach, 2005], are based on formal semantics and transition rules. The possible states of the organization are described by state transitions. The state description captures the whole system; they include the organizational structure as well as the internal knowledge of the agents. Matson [Matson and DeLoach, 2005] includes internal knowledge of the agents within the specification. The agents are restricted by the organizational specification. According to several definitions, autonomous agents should have control over their internal state. Therefore, the autonomy requirement is not fully met in this case.

Electronic institutions as described by Esteva et al. [Esteva et al., 2001] and Garcia et al. [Garca-Camino et al., 2007] are norm-based coordination models for

open multi-agent systems. The norms are specified in a multi-agent middleware, to regulate agents' actions. The institution actively interacts with the agents. The same holds for the electronic institutions as for the Moise+ model: the middleware checks whether an agent choices are allowed. Therefore, the agents have their local reasoning, but are not fully autonomous in controling their behavior.

2.5.4 Summarizing

The discussion above shows that there is a tight relationship between autonomy and coordination. Choices on the coordination mechanism can lead to restrictions on agent autonomy. Concerning autonomy, a good starting point is that an organization is defined separately from the internals of the agents. The organizational rules should be used as guideline for the agent's decisions, but the organization cannot enforce behavior. Furthermore, an abstract way of coordination guarantees that the actors maintain freedom in choosing on their own actions. As a consequence, the actors can be held responsible for their own behavior. This approach to coordination fits well with our research goals concerning autonomy.

However, the guarantee of agent autonomy is a design choice. There are several implementations of coordination mechanisms that allow some degree of enforcement or regimentation of agent behavior by the organization. The optimal solution depends on the application.

2.6 Conclusion

In Chapter 1 we argued that autonomy is a key concept in multi-agent systems. In the philosophical context, autonomy is taken as basis for responsibility. We want to adopt this view to agent systems. Although autonomy is often used to define agents, there is no single definition of agent autonomy and adjustable autonomy.

In agent research, several approaches of autonomy and adjustable autonomy have been discussed. Most of them focus on the process of action selection. We have argued that the process of controling the internal state requires more attention in autonomy research. We have defined autonomy as *having control over external influences on the decision-making process.* Adjustable autonomy in our context means: *dynamically dealing with external influences on the decision-making process based on internal motivations.* In contrast to some other works, we give the agent control over its autonomy, and therefore it is a form of self-adjustable autonomy. Adjustable autonomy gives the agent the opportunity to change its attitude with respect to the environment and to other agents. It can be used to achieve dynamic coordination.

In section 2.4, we have presented an abstract reasoning model that guarantees agent autonomy. We have discussed general reasoning models for reactive reasoning and BDI-reasoning in the context of autonomy. In section 2.5 we showed that our view on agent autonomy guarantees the distributed feature of a multi-

agent system and allows for coordination. Furthermore, we discussed the relation between autonomy and existing coordination mechanisms. Coordination mechanisms have consequences for the autonomy of actors in a multi-agent system. By choosing an abstract coordination model that is defined separately from the internals of the actors, the autonomy of the actors is facilitated and the actors can be held responsible for their own actions.

Next, we will develop a reasoning model for agents more in detail that matches our view on autonomy. This is the focus of Chapter 3 of this thesis.

Chapter 3

Autonomy in the Reasoning Model

As discussed in Chapter 2, researchers have developed several models of autonomy and adjustable autonomy. Many papers have studied a way to include adjustable autonomy in the decision-making process of an agent. However, there is no single definition of autonomy. All approaches point to different parts of the decision-making process and have different notions of adjustable autonomy accordingly. Sometimes the level of autonomy is controlled outside of the agent, or when self-adjustable autonomy is allowed, it is unclear how it is included in the reasoning model. In the previous chapter, we have presented an overview of related literature on agent autonomy and adjustable autonomy.

We start from the assumption that an autonomous agent has control over external influences, following the definition of autonomy from Chapter 2. In relation with agent decision-making this means that the agent has control over how external input influences the mental state of the agent. In our definition autonomy is having control over external influences and adjustable autonomy is dynamically dealing with external influences. Concerning coordination, an agent allows its decision making to be influenced when coordination is required. The agent does not necessarily decide upon an autonomy level, but the autonomy level follows from the way the agent handles external input.

In this chapter, we propose a reasoning model that guarantees agent autonomy and at the same time allows coordination. We motivate our choice for BDI-reasoning models and we extend the BDI-reasoning model with a component for controling external influences. We propose the use of reasoning rules to process incoming events. The set of active event-processing rules determines the attitude of the agent with respect to the outside world. With an experiment we show how attitudes can be specified.

Further in the chapter, we discuss different types of external influences. We present general heuristics process external events. To have control over external influences implies that the agent can reason about the heuristics.

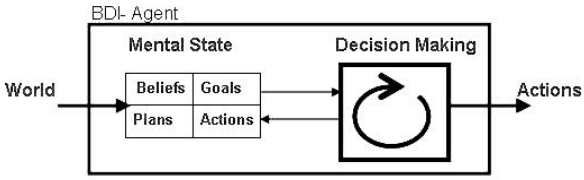


Figure 3.1: *An abstract reasoning model of a BDI-agent.*

3.1 BDI-Reasoning

In this section we will take a closer look on autonomy in BDI-reasoning, and particularly we investigate whether agents are aware of the external influences to which they are subject. We discuss whether BDI agents have control over external influences on their decision-making process.

The Belief-Desire-Intention (BDI) model for agent reasoning as introduced by Rao and Georgeff [Rao and Georgeff, 1995] is based on a model to describe human reasoning as developed by Bratman [Bratman, 1987]. The BDI-model describes logic-based reasoning at an abstract level. The concepts that are used for the reasoning are semantic concepts. This makes BDI-reasoning comparable to human reasoning. Furthermore, when it comes to social capabilities of the agent, communication and interaction can be described in an intuitive way using speech acts. The agents reason with longer-term goals, which enables pro-activeness and guarantees consistency over time. The above arguments support the choice for a BDI-reasoning model in our research, as we expect agents to have social skills that interact with each other and with humans.

3.1.1 BDI Implementations

We can view a BDI-agent in an abstract way as a mental state and a reasoning process. The mental state captures the beliefs, desires and intentions of the agent, the reasoning process decides upon actions based on the mental state, Figure 3.1.

The BDI approach is a well known model for deliberative agents. The mental state of the agent contains beliefs, desires and intentions. Beliefs represent the current world as it is interpreted by the agent. Desires describe future states of the world that an agent wants to reach, so called goal states. The intentions are a subset of the desires and contain the goal states to which the agent commits itself to reach. Furthermore, plans are generated based on the mental state of the agent, which specify how to reach the intentional states. Such a plan contains sub-plans and actions, the execution of which is expected to result in the goals.

The mental state is updated by the reasoning process via internal actions. The mental state can also be updated by external influences. Sometimes this concerns active observations; observations that are deliberately selected by the reasoning

process. In most situations it concerns passive observations; events that reach the agent from the environment such as messages. The latter is often the case when dealing with situated agents, where stimuli from the environment continuously reach the agent.

A BDI-deliberation cycle describes in an abstract way the reasoning functions that need to be performed; generate optional plans from beliefs and desires, select plans from the optional plans that are 'promoted' to the set of intentions, and execute those intentional plans. In all of the functions the mental state of the agent determines the outcome. There are several models of BDI-deliberation processes, which give their own interpretations on some of the parts of the deliberation. We will briefly describe two models; first the model for BDI agents as it is described in [Rao and Georgeff, 1995], that has been used e.g. for the PRS agent-model and the Jason platform [Bordini et al., 2005] and secondly the 3APL interpretation of BDI-agents [Hindriks et al., 1999] [Dastani et al., 2004b] and its follow up 2APL. We will point out some of the interpretation differences.

The PRS-Model

The deliberation process of the PRS-model as first described in [Rao and Georgeff, 1995] is given in pseudo code below. It describes the plan generation, selection and execution. This model introduces *events* as an additional construct. Events are used for describing changes in the agent world, which can be internal as well as external changes. The optional plans that are generated from the events contain actions to reach a certain goal. Deliberation about the options leads to the intended plans.

```
initialize-state();
repeat
    options := option-generator(event-queue);
    selected-options := deliberate(options);
    update-intentions(selected-options);
    execute();
    get-new-external-events();
    drop-successful-attitudes();
    drop-impossible-attitudes();
end repeat
```

In this agent model desires are represented implicitly. Plans are derived from the agents mental state (i.e. beliefs and events), by following pre-defined reasoning rules. The reasoning rules implicitly contain desires. The PRS-model has been used as basis for the Jason platform [Bordini et al., 2005]. Jason supports the BDI-programming language AgentSpeak(L) [Rao and Georgeff, 1995] to program BDI agents.

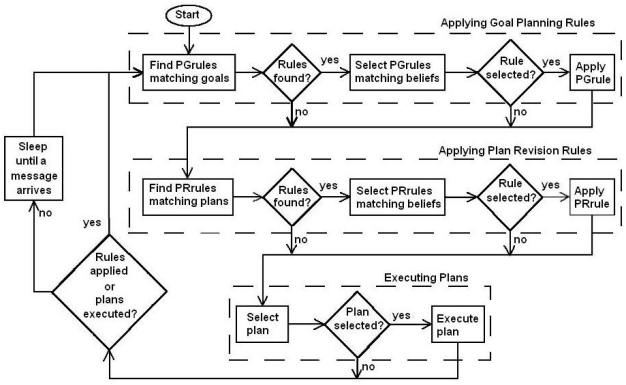


Figure 3.2: *Deliberation cycle of 3APL*

The 3APL-model

The 3APL deliberation-cycle [Dastani] is presented in figure 3.2. The general procedure is the same. First plans are generated and selected from reasoning rules using the mental state of the agent and then the plans are executed. Two types of reasoning rules are explicitly addressed in the 3APL deliberation; goal-planning rules and plan-revision rules. Goal-planning rules generate plans from goals in order to reach the goals. Plan-revision rules allow for the possibility to revise an intentional plan. Applicable rules are selected by matching them with the goals from the agent's goal base and the beliefs from the belief base. From the applicable rules, intended plans are selected.

In 3APL goals are similar to desires from the previous model and in this paper we use both terms. In contrast to the previous model, the 3APL implementation includes an explicit goal representation using a goal base [Dastani et al., 2004b]. Another difference is the handling of internal and external changes of the agent world. In the 3APL deliberation the connection to the external world can be made via messages. The deliberation cycle transforms a received message automatically into a belief.

3.1.2 Updating the Mental State

Our aim is to identify the effect of influences from other agents and the environment on the agent's reasoning process. We need to know how the agent's mental model is updated and how actions are selected during the deliberation. Some choices that are made in the deliberation cycle have effect on the updating of the agent's mental state. Therefore, we make some general observations.

Starting with *beliefs*, in standard BDI models they are represented explicitly in a belief base. Updating beliefs can take place by a specific function in the deliberation cycle, for example the message receiver in the 3APL implementation.

Another way is by executing internal actions that are part of a plan. The former can be seen as an *unconscious* process that the agent undergoes, whereas the last option takes place by executing an action that has been chosen deliberately.

The update process of goals (or desires) is comparable to that of beliefs, but it is not common to let an agent unconsciously adopt a new goal. Goal adoption should take place by executing an internal action that has been chosen deliberately. Dropping a goal that has been achieved or that is impossible to reach is sometimes done in the deliberation cycle, e.g. via the *drop-successful-attitudes()*- and *drop-impossible-attitudes()*-function in the PRS-model. It can be argued that goals can be subject to reasoning as well. A model for goal revision using reasoning rules is proposed in [da Costa Pereira et al., 2006]. This model enables more dynamic goal-updates.

In the general BDI model intentions are added by committing to a plan following pre-defined reasoning rules. After execution, the plan can be removed from the intention set automatically. It is possible to extend the deliberation cycle with other functionalities that can modify an agent's intentions, as is introduced by [Riemsdijk et al., 2005] and shown in the deliberation cycle of 3APL in figure 3.2. Reasoning rules can be provided that allow the agent to revise its plans.

As observed, updating the mental state of the agents can be separated in an *unconscious* and a *conscious* part. The former depends on the choice and the implementation of the deliberation process where pre-determined mental updates take place. The conscious updating is caused by the reasoning of the agent itself, i.e. by its reasoning rules and its capabilities. By using them the agent can update its own mental state deliberately.

3.1.3 Event Processing in a 3APL-Agent

Here, we give some real examples of how agents can be influenced by external events. The examples illustrate that there is not always deliberate reasoning about adoption of beliefs and goals, and as such an agent is sensitive for manipulation. We use 3APL programs to describe the examples.

A 3APL agent is constructed with six bases containing capabilities (i.e. internal actions), beliefs, goals, plans, plan-generation rules and plan-revision rules. After defining the initial state, the agent can run the deliberation cycle from figure 3.2. In a 3APL agent, the capabilities, plan-generation and plan-revision bases are static. The belief-, goal- and plan bases are dynamic and are continuously updated in the deliberation process.

3APL has two connections to the outside world; via messages and via external actions. Receiving a message produces a belief of the form `received(Sender, Performative, Content)`. Consider the example below; when the agent receives a message, it adds one to its counter. This is a form of passive observation, which implies that an agent cannot reason about adopting the resulting belief. Therefore, the 3APL deliberation cycle provides a way for influencing an agent

by sending messages to this agent. When its goal is *count*() The agent activates the action *AddOne*() every time it receives a message.

```
PROGRAM "counter"
CAPABILITIES:
  { counter(X) } AddOne() { NOT counter(X) AND
                            counter(X+1) }
BELIEFBASE:
  counter(0)
GOALBASE:
  count()
REASONING-RULES:
  count() <- received(You,Performative,Content) |
  { AddOne() }
```

Similar, but slightly different is the case when the head of the reasoning-rule is empty. In that case, the agent is not dependent on its goals to activate the plan. Therefore, it adds one to its counter after any message at any time. This is the case when the reasoning rule of the *counter*-agent is replaced by the following.

```
REASONING-RULES:
    <- received(You,Performative,Content) |
  { AddOne() }
```

The programmer is free to add situational constraints to the processing of a message by extending the reasoning rule. The third example shows a rule with which the agent only activates the action *AddOne*() when the sender of the message is trusted.

```
REASONING-RULES:
    <- received(You,Performative,Content) |
  {
    IF trusted(You) THEN
      { AddOne() }
  }
```

The second way in 3APL to connect to the outside world is by external actions. An external action calls a function in an external module. It has the form `External("ModuleName", function(), RESULT)`. The result of the external function is given back to the agent as a variable, *RESULTS* in the case, in the reasoning rule. This can be used as a way of active observation; the observe action is selected deliberately and the agent can choose to adopt the results in its belief base. Consider the following reasoning rule. The agent chooses to do the observation and adopts the result. Again, situational constraints can be added within the reasoning rule.

```
lookAround() <- TRUE |
{
  External("World", lookAround(), RESULT )
  AdoptBelief(RESULT)
}
```

3.1.4 Are BDI-Agents Autonomous?

The definition of autonomy as presented in Chapter 2 requires that an agent has control over external influences. Can we say that the agent is autonomous based on the above examples? Does the agent control the external influences? We want to argue that this is not guaranteed to be the case. The agent has reasoning rules with which it processes messages and observations. However, the reasoning rules of the agent are static. That means that the agent cannot actively choose the way it processes the events.

As the examples show, it is possible to program the agent such that it has sophisticated event handling. All situational constraints can be included in the reasoning rules. Therefore it is possible to program the agent such that it does not experience unwanted influences. However, the knowledge for influence control is all part of goal-directed reasoning rules. Therefore, sophisticated event handling will lead to complex reasoning rules. The rules no longer focus on planning to reach the goals only, whereas that is what they are meant for. They need to deal with handling external influences as well.

We believe that the agent becomes more autonomous when it is designed with a separate module for influence control, as introduced in Chapter 2. This module can contain reasoning rules that primarily focus on processing external events. Furthermore, we propose a mechanism to enable dynamics in the reasoning rules, such that the agent can adapt the influence handling at runtime. In the next section we present the module for influence control in further detail.

3.2 Extending the Reasoning Model with Event Processing

In this chapter, we present a reasoning model for agents that enables the agent to control external influences. The level of autonomy of the agent depends on external influences on the reasoning process. In the reasoning-process we distinguish a module for event processing and a module for decision making. The event-processing module gives the opportunity to the agent to deal with external influences in an explicit way. The decision-making module focuses on the decision on action. We describe the implementation of the two parts.

We propose add a component for event processing to the BDI reasoning-model. Figure 3.3 shows the result in a graphical way. Here, the component for influence control determines the effect of external input on the mental state of the agent. Subsequently, the mental state is used to determine the actions of the agent.

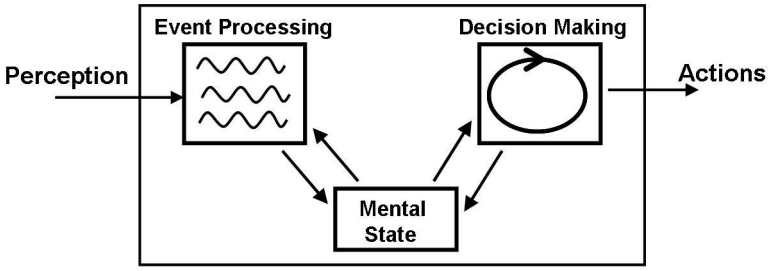


Figure 3.3: *Abstract model of BDI reasoning with an event-processing module*

controling external influences is an internal process, just as action selection. By explicitly adding influence control to the agent's reasoning-model, we guarantee the autonomy of the BDI-agent.

3.2.1 Module for Decision Making

In the decision-making module the agent will decide upon the next action. We adopt regular BDI-reasoning for this module. In the work presented here, we use the 3APL implementation of BDI-reasoning to illustrate our examples. However, any BDI implementation can be used here.

3APL [Dastani et al., 2004b], [Hindriks et al., 1999] provides the designer with a formalized programming language which is designed for BDI-agent programming. A 3APL agent uses reasoning rules to create plans to reach a certain goal. Such reasoning rules have the following form:

```
<HEAD> <- <GUARD> | <BODY>
```

The head of a rule should match the goals of an agent. The guard should match the beliefs of the agent. The body of the agent contains sets of actions. If head and body match, the agent can commit to the plan in the body and start to execute it. The decision-making process uses these type of reasoning rules together with the beliefs and goals of the agent to determine its actions. Figure 3.2 shows the deliberation process of a 3APL agent.

3.2.2 Module for Influence Control

The module for influence control precedes the decision making. External events are processed here. The module determines how external events influence the decision-making process. Will the agent follow the commands or requests, or will it only reason with its own goals? Does the agent adopt information from another agent, or does it use its own observations? We show how we can implement reasoning rules that enable the agent to control external influences.

We propose to implement this module using similar reasoning rules as in the decision-making module. This gives the opportunity to reason with the knowledge from the agent's mental state. The reasoning rules have the form:

```
<HEAD> <- <GUARD> | <BODY>
```

The head matches the incoming events. The guard describes situational constraints, and should match the beliefs and goals of the agent. The body of the rule specifies the effect on the mental state.

External events can be an agent's observations or messages from other agents. Therefore, the event-processing module has two types of reasoning rules; rules that handle observations and rules that handle messages.

Handle Observations

Reasoning rules for observation handling specify how incoming observations are processed. For example, an observation can be adopted into the belief base without further constraints.

```
observation(X) <- TRUE | UpdateBeliefs(X)
```

It is possible too add more rules that distinguish between different situations. In the guard of the rule constraints can be specified, as the guard has to match with the beliefs of the agent. The guard can contain any knowledge that is available in the belief base. Imagine that an agent wants to focus on specific information and this information is described by a relevance construct. The following rule can be specified, that limits the processed observations to those that are relevant:

```
observation(X) <- relevant(X) | UpdateBeliefs(X)
```

Handle Messages

Agents can receive messages from other agents. An agent can be programmed to handle messages in different ways by adding the same types of rules. We assume that messages are constructed containing a *sender*, a *performative* and some *content*. A general rule to adopt message content to the belief base looks like the following.

```
message(SENDER, PERFORMATIVE, CONTENT) <- TRUE |
  UpdateBeliefs(CONTENT)
```

The performative of a message contains information about the intention of the message, and of the content of a message. Typical performatives are *inform* to inform the recipient about something, and *request*, which is used to ask an agent to do something. The message generally contains information or a task respectively. In the event-processing rules we can specify the performative of a message:

```
message(S, inform, X) <- TRUE | UpdateBeliefs(X)
message(S, request, X) <- TRUE | AddGoal(X)
```

The guard of the reasoning rule can contain any knowledge from the agent's mental state. Therefore, the guard can distinguish messages by sender and content. The examples below show how this distinctive feature can be used in reasoning rules:

```
message(S, P, X) <- trusted(S) | UpdateBeliefs(X)
message(S, P, X) <- relevant(X) | UpdateBeliefs(X)
```

3.2.3 Effects on the Mental State

The result of an event-processing rule is an internal action of the agent, which specifies the effect of the event on the mental state of the agent. The examples of reasoning rules as given above had two possible effects on the mental state; *UpdateBeliefs* and *AddGoal*. However, these two are not all possible results of event-processing rules. The possibilities depend on how the mental state is constructed.

A belief update is not necessarily a straight-forward process. Adopting a belief can lead to inconsistent beliefs. If this is not allowed by the agent architecture, other beliefs might need to be removed as a consequence. Another issue is whether the belief base contains negations or follows the closed world assumption. Furthermore, some BDI-agent models allow deduction rules in the belief base. A 3APL belief base, for example, can contain Prolog deduction besides simple facts. At this point, we specify belief updates in a general way with the internal action *UpdateBeliefs(X)*, which implies that the belief base updates with the fact X.

Concerning goals, we used *AddGoal* in the examples. This is the consequence of accepting a request, for example. In 3APL, the goals do not need to be consistent with each other. Another internal action can be to drop a goal, as response to a prohibition, for example. Therefore, we define the action *RemoveGoal*.

Another option is to specify rules that result in ignoring the external event. The result of such rule is actually to do nothing, whereas it takes effort. The same result might be reached by leaving the rule out. However, with an ignore-rule the agent makes the choice to do nothing explicitly. Specifying to ignore an event could be useful to indicate exceptions. We give example rules for when information is irrelevant, or when the agent is busy:

```
observation(X) <- NOT relevant(X) | ignoreEvent()
message(S, P, X) <- busy() | ignoreEvent()
```

Table 3.1 lists the internal actions and the corresponding effect on the mental state. For the 3APL agent that we use, we propose the four internal actions as result the event-processing rules. In the examples and experiments presented in this thesis, we work with this set of internal actions. However, as we stated, this

set does not necessarily contain all possible results of event-processing rules. For the 3APL agent that we use, the presented four are sufficient to demonstrate control over external influences.

Table 3.1: *Effects on the mental state*

Internal action	Effect on mental state
UpdateBeliefs(X)	Update belief base with belief X
AddGoal(X)	Add goal X to goal base
RemoveGoal(X)	Remove goal X from goal base
IgnoreEvent()	Ignore the external event, no effect on mental state

In the next section we demonstrate how we can create different levels of autonomy via event-processing rules. The event-processing rules specify the attitude of the agent towards the outer world.

3.3 Influence-Based Attitudes

The set of active event-processing rules of the agent determines how the agent perceives the world and how the decision-making process is influenced by external factors. In this section, discuss different types of inter-agent influences and we demonstrate how we can create different attitudes by specifying the event-processing rules. In a multi-agent environment, the attitudes of the agents have consequences for the group behavior. We show how we can create different coordination styles by varying the attitudes of the agents.

3.3.1 Types of Inter-Agent Influences

When we consider a multi-agent system, there are several ways in which agents can influence each other. Following Falcone [Falcone and Castelfranchi, 2001] and Barber, [Barber et al., 2000], we distinguish three types of inter-agent influences: environmental influence, belief influence and goal influence. We will describe the three types of influences informally.

Environmental Influence

An agent is situated in an external environment. Environmental influence occurs when an agent observes environmental aspects and uses the observations to reason with. In order to be open for environmental influences an agent needs the capability to observe the external world. Furthermore, the observations need to be transformed into constructs the agent reasons with. For example, if an agent

that reasons about the world in terms of beliefs it would need sensors that transform external input from the world to beliefs. The observation results in internal action where the agent adopts the observation into its belief base.

```
observation(X) <- TRUE | UpdateBeliefs(X)
```

To be open for environmental influences, an agent needs to be able to observe the external environment and transfer the observation to beliefs. Influencing an agent via environmental modification only has an effect if the beliefs that are obtained from the observation are relevant in some way for the agent's decision-making process. From the perspective of the agent experiencing the influence it is useful as well to know whether observations are relevant for its decision making or not.

Belief Influence

We say that an agent's internal world model is captured in its beliefs. Belief influence occurs when an agent uses information in its reasoning process that is taken from a communicative act from another agent. Assuming that communication is done by sending messages, the agent should have the ability to receive messages and it should process the message such that the message content becomes available for reasoning. In order to be open for belief influence an agent that reasons with beliefs needs reasoning rules to process the message:

```
message(SENDER, inform, X) <- TRUE | UpdateBeliefs(X)
```

To be open for influences via beliefs an agent must have the capability to communicate and to extract information from the communicative act. Assuming that an agent has those capabilities, new beliefs will be created. The influence on an agent's reasoning process will only take place if the created beliefs are actively used in the reasoning process. Again, knowledge about relevance of the message content might be useful for an agent for processing it.

Goal Influence

Assuming that an agent is pro-active, we can expect that an agent has one or more goals that it actively pursues. Goal influence occurs when an agent considers goals that come from a communicative act from another agent. This means that the agent adds a goal to its goal base that is taken from an incoming message from another agent. This type of influence occurs for example in hierarchical relations, where one agent can tell another what task should be fulfilled. In order to be open for goal influences, the agent needs the ability to process a message and extract a goal from it, for example:

```
message(SENDER, request, TASK) <- TRUE | AddGoal(TASK)
```

Table 3.2: *Options for event processing.*

Influence Type	**Trigger**	**Adopting Position**	**Rejecting Position**
Environmental	Observation	Self-reliant	Non-self-reliant
Belief	Inform Message	Trusting	Non-trusting
Goal	Request Message	Cooperative	Non-cooperative

The agent adds the goal to its goal base. This means that an agent pursues to reach the goal. In the context of goals and tasks, it is useful for the agent to know whether it has capabilities to achieve the goal or not.

Manipulation via Influences

Openness to the influence types provides the agent with certain qualities. In order to act in a dynamic environment an agent needs to adapt to environmental changes and therefore it needs to be open for environmental influences. On the other hand, the agent might become sensitive for manipulation via the environment. That can be the case when influences on the mental state affect the decision-making process. If it is predictable how an agent will react on certain environmental changes, the agent can be manipulated causing those changes on purpose.

The same holds for the other influence types. In order to show social capabilities it needs to be able to react on other agents' messages. Belief influence is useful for information sharing and goal influence can be useful in relational aspects in a multi-agent system. Both reach the agent by communication. Manipulation becomes possible if it is predictable how messages are processed and how the content is used in the decision-making process. In order to prevent manipulation it is important for an agent to be able to reason about external influences.

3.3.2 Basic Attitudes

We will relate inter-agent influences to attitudes of how agents perceive the world. The attitudes are directly related to levels of autonomy of the decision-making process. For all three types of influences the agent can choose to adopt or reject them, as listed in Table 3.2. This results in a position with respect to an influence-type.

From the three influence types and the two processing options, we can construct eight different attitudes. We present three basic ones:

- *Non-social*: An agent with a non-social attitude does not interact with other agents. It adopts own observations, but messages from others are ignored. Therefore, the goal/task determination is free from external influences. The

agent creates and selects its own goals and plans. Influence via environmental modification is still possible; it is possible to influence the agent's behavior by manipulating the environment. The following event-processing rules for messages and observations are active:

```
observation(X) <- TRUE | UpdateBeliefs(X)
message(Sender, Performative, X) <- TRUE |
    IgnoreEvent()
```

- *Self-reliant/trusting*: A self-reliant, trusting agent will process messages from others and believe the content. Its beliefs are influenced by others. The agent adopts own observations as well. The agent determines its goals and tasks by itself. We have implemented the *self-reliant, trusting* attitude by the following event-processing rules:

```
observation(X) <- TRUE | UpdateBeliefs(X)
message(Sender, inform, X) <- TRUE |
    UpdateBeliefs(X)
message(Sender, request, X) <- TRUE | IgnoreEvent()
```

- *Self-reliant/cooperative*: If agent A is cooperative with respect to agent B, it will do what agent B asks for, without considering other options. Its tasks and goals are determined by agent B. Agent B can send a request message to agent A. A processes the message by adding the request to its goal base. The agent adopts own observations in order to create an own world perspective. The following rules specify the self-reliant, cooperative attitude:

```
observation(X) <- TRUE | UpdateBeliefs(X)
message(S, inform, X) <- TRUE | IgnoreEvent()
message(S, request, X) <- TRUE | AddGoal(X)
```

These are three examples of the eight possible attitudes. Of course, it is possible to construct more complex profiles by combining of specifying the event-processing rules differently. All the attitudes can be implemented using our module for influences control. In some sense the attitudes determine the basic interaction styles between agents embedded internally in the deliberation process. The interaction styles have consequences for coordinated behavior in organizations. Next, we discuss what types of influences exist between agents in a multi-agent system.

3.4 Coordination Experiment

In this section we introduce an experiment demonstrate how to implement influence-based attitudes and to illustrate the effect of different attitudes on organizational performance. We have defined an agent organization, in which the agents can act following different attitudes. We want to observe properties of the organization when it is populated by different types of agents.

3.4.1 Organizational Description

The general setting is a fire brigade organization. There is a world with fires, firefighters and a coordinator. The aim of the organization is to extinguish the fires as fast as possible. In the agent organization two roles have to be fulfilled: Coordinator and Firefighter.

- *Coordinator*: Plan which fire is to be extinguished by which fireman, and send commands to the firefighters.
- *Firefighter*: Move around randomly to look for a fire, select a fire and extinguish the selected fire.

The agents playing the roles have been implemented following the reasoning model described in Section 3.2. The modules for influence control and for decision making have been implemented sequentially. We used the 3APL-reasoning mechanism for the decision-making module. Goals, beliefs, plans, and basic actions have been made explicit.

In our simulation, firefighters are situated in an environment, where fires can pop up. A firefighter can move to a fire and extinguish it. Fires are growing gradually in time, except for when they are being extinguished, then the fire size decreases and they will disappear. The firefighters have a limited view. The coordinator agent has a global view, it can see all fires. The only action the coordinator can take is sending commands to the firefighters, telling them which fire they should extinguish. The coordinator has one handicap, which is that he can send only one message per time interval.

We have constructed three different types of firefighters based on their attitudes: *non-social*, *self-reliant/trusting* and *self-reliant/cooperative*. We created the profiles by activating or de-activating the event-processing rules as listed in Section 3.3. The agents vary on how they process messages from others. In all attitudes influence via environmental modification is possible, so the observations are always adopted into the belief base.

- *Non-social*: Non-social firefighters observe the fires and select the fire they want to extinguish all by themselves. Influence by environmental modification is possible, for example when a firefighter observes that a fire is getting smaller, because another firefighter is extinguishing it. The agents do not process any message from each other or from the coordinator agent.

Table 3.3: *Reasoning profiles of firefighters in their relation with the other agents. Three different organizations.*

organizational characteristic	**Attitude firefighter regarding firefighters**	**Attitude firefighter regarding coordinator**
Non-social	Non-social	Non-social
Trusting	Self-reliant/Trusting	Non-social
Cooperative	Non-social	Self-reliant/Cooperative

- *Self-reliant/trusting*: Self-reliant/trusting firefighters are communicating-with the other agents. If they are busy extinguishing a fire and meanwhile observe another fire, they send a message to the other firefighters to inform them about the fire. The receiving agent processes the message. The information of the particular fire is added to the beliefs. There is belief influence by sending messages to each other informing them about fires. There is still influence by environmental modification by other agents extinguishing fires.

- *Self-reliant/cooperative*: Self-reliant/cooperative firefighters are commanded by another agent, who tells them which fire they have to extinguish. They do not take initiative by themselves. There is influence on task determination by adopting the requests from the coordinator agent. The firefighter has an obedience relation with the coordinator. No matter what goal it has, if received a request from the coordinator to extinguish *FireX*, it will adopt the goal *ExtinguishFire(FireX)*. There is still influence by environmental modification by other agents extinguishing fires.

In our experiment we used an organization consisting of two firefighters and one coordinator. We have created the organizational composition by varying the event-processing rules of the firefighters. We have created three different organizations, as shown in Table 3.3. In the *non-social* organization the firefighter agents ignore all other agents; their profile with respect all other agents is *non-social*. In the *trusting* organization the firefighters trust each other, but ignore the coordinator agent. In the *cooperative* organization the firefighters are cooperative to the coordinator and ignore each other. Note that the difference between the organizations has been created purely by constructing different attitudes for the firefighter agents with respect to other agents in the organization.

We evaluated the performance of the organizations by measuring the time it took to extinguish all the fires. To investigate the performance of the different organizations, we used environments with different characteristics. The firefighters started in a world containing five fires. In one situation the fires were spread randomly over the field. In the second situation the five fires were clustered in a group. A graphical representation of the experiments is shown in Figure 3.4 for the environment with randomly spread fires and Figure 3.5 for the environment

with a group of clustered fires.

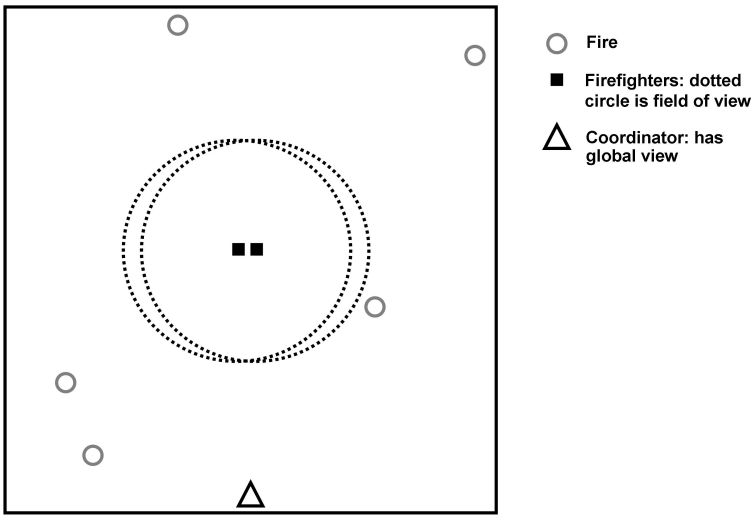


Figure 3.4: *Example environment with randomly spread fires.*

3.4.2 Results and Discussion

We have run several simulations of the six situations; three different organizations each performing the task in two environments. The organizations differ only in the attitude of the firefighters. The environments differ in the distribution of fires. In one run, the firefighter organization had to extinguish all five fires. The performance measure equals the time it took the organization to fulfill this task. Figures 3.6 and 3.7 show the average extinguish times and the corresponding standard deviation for 100 runs per situation in the random environment and the clustered environment respectively.

Comparing the results of the random and clustered environments, we see the biggest difference in performance for the *trusting* firefighters. They perform worse in the situation of randomly spread fires. This can be explained by the fact that they will only view the fires one by one. They are not able to take advantage of their communication, since in their behavior it was specified that they would inform each other in the situation when they saw more then one fire. They have to search for each fire individually, which explains the growing standard deviation. As a result they perform comparable with the *non-social* organization.

If we consider the only clustered environment, the results in Fig. 3.7 show a difference in performance of all three organizations. The *cooperative* firefighters perform best. They get orders from the coordinator agent, which has a global

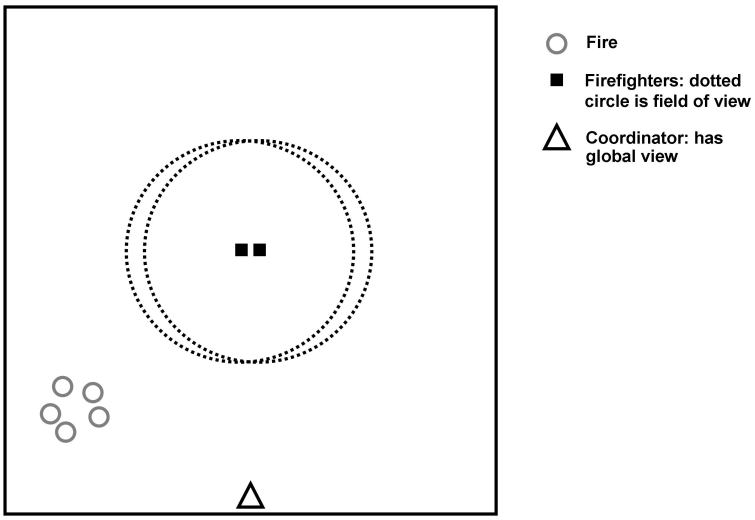


Figure 3.5: *Example environment with a group of clustered fires.*

view, so all fires are known from the start and the coordinator just sends the firefighters to the right places. In the *non-social* and *trusting* organizations the firefighters do not process messages from the coordinator agent. They have only a local view, so they first need to look for the fires. When one firefighter has found a fire, he will see the other fires as well in the clustered situation. The *trusting* firefighters exploit this knowledge by telling each other about the fires, so the second firefighter immediately joins to help. In the *non-social* organization the firefighters do not have this ability of information sharing. As an illustrative example we show the results of a typical run in the clustered situation in Fig. 3.8. The x-axis shows the time and the y-axis shows the total fire size in the environment for all three organizational types. It is visible that once the *trusting* firefighters have found the first fire, the fire size decreases faster than for the others. In the *cooperative* organization the fire size decreases most constantly.

In the above presented simulations, the *cooperative* organization outperformed the other two on average. The organization uses the global view of the coordinator agent. The firefighters follow the orders of the coordinator and it does not really matter how the fires are distributed. However, this type of organization has limitations as well. The organization works by a centralized approach and is very sensitive to the performance of the coordinator agent. The restriction on the number of messages that the coordinator agent can send, is not a big issue in the small organization we use here, but it will be in larger organizations. Another problem is that the observations of the coordinator agent play a very important role, since he determines which fire is to be extinguished by which firefighter.

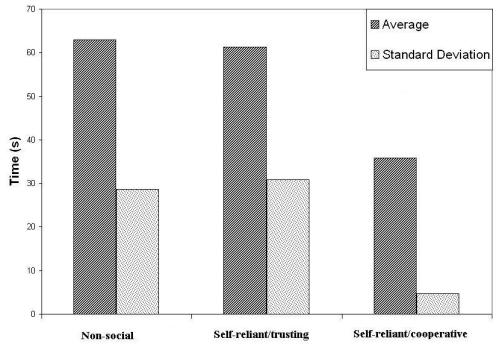


Figure 3.6: *Extinguish times over 100 runs; average and standard deviation. Random fire distribution.*

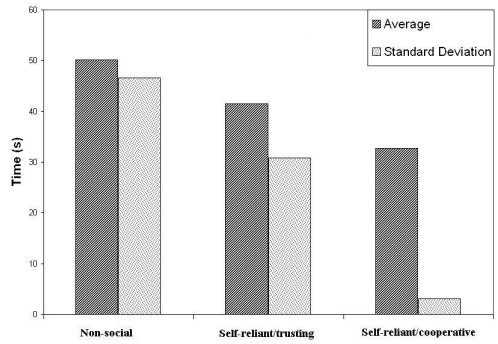


Figure 3.7: *Extinguish times over 100 runs; average and standard deviation. Clustered fires.*

Failure of the observations of the coordinator has big consequences.

All results of our experiment can be explained by analyzing the information flow in the agent organization. We want to point out that the goal of this experiment was mainly a proof of concept. We have defined different organizations by making the agents reason based on different influenced-based attitudes with respect to the other agents. In our experiment the organization was still static. By defining the attitudes internally in the agents within the reasoning model we also create possibilities for switching between autonomy levels at runtime, and therewith allow dynamic organizations.

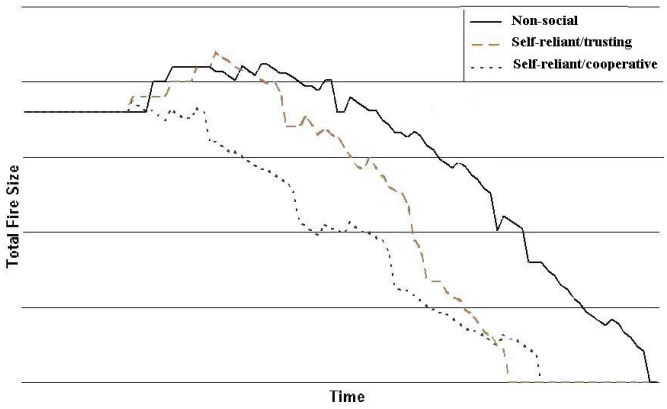


Figure 3.8: *Total fire size over time, fires are clustered.*

3.5 Dynamic Coordination Experiment

In the previous section, we introduced an experiment to illustrate the concept of autonomy levels in agent decision-making. We have defined an agent organization, in which the agents have to coordinate their actions. We created organizations with different features by giving the agents different attitudes. We observed that performance of the organization depends on the attitudes of agents.

In this section, we motivate the need for adjustable autonomy in order to achieve the required type of coordination in a dynamic environment. We use the same experimental environment to test the characteristics of different types of coordination to environmental characteristics. The firefighter organization performs a simple coordination task. We have constructed different scenarios, which contain situational features that can reveal the strong and the weak points of each coordination mechanism.

3.5.1 Coordination Mechanisms

The basic setting is again the firefighter organization. The organization operates in a world where fires appear that need to be extinguished as fast as possible. In the organization we define two roles; *coordinator* and *firefighter*. The coordinator makes a global plan and tells the firefighters which fire they should extinguish. Therefore, the coordinator has a global view of the world. The firefighters perform the actual tasks in the world; they move to a fire location and extinguish the fires. They have only local view. There is a hierarchical relation between the two roles, the coordinator is superior of the firefighters and can send orders to the firefighters. He can tell which fire they have to extinguish.

We want to show different forms of coordination within this organization. In our implementation we achieve this by changing the autonomy level of the decision-making process of the firefighters. We have created two types of firefighters; *self-reliant/cooperative* agents that follow the orders of their superior and thus have no decision-making autonomy and *non-social* agents that ignore their superior and make their decisions only based on local observations. The coordination mechanisms that follow from those firefighter attitudes are *emergent coordination* and *explicit coordination*:

- *Emergent coordination*: non-social firefighters, choices are made based on local information
- *Explicit coordination*: cooperative firefighters, choices are made based on global information

We want to evaluate the performance of the organization. In our experiment we can measure the time it takes to extinguish the fires for each of the coordination types. The best organizational performance has the lowest score.

3.5.2 Scenarios

For this experiment, we constructed three scenarios, each with specific environmental features. There is no randomness involved. The goal is to show that there is a relation between environmental features and the performance of coordination mechanisms. In all three scenarios, the organization contains one coordinator and four firefighters. The starting position of the firefighters in the world is equally distributed. We have one *standard scenario*, scenario A, in order to test whether both coordination types perform equally well. In this scenario four fires are distributed equally over the world. The start situation of scenario A is shown in Fig. 3.9.

Two other scenarios have been created that make this situation more complex. They contain the features that also occur in real world situations. In our first experiment, the *cooperative* organization performed best. However, as we argued, the organization is highly dependent on the reliability of information of the coordinator. In scenario B we will investigate this feature. Scenario B is a setting where the fires are distributed equally over the world, but the coordinator has no perfect view. At the start of the scenario, the coordinator can only see half of the fires. The other half of the fires are there, but they are only observable at the local level. After some time, they become known to the coordinator via communication with the firefighters. Figure 3.10 shows the starting situation of Scenario B.

The third scenario, Scenario C, reflects a situation where the fires are not distributed equally, such that some firefighters do not observe any fires, whereas others observe several fires. Figure 3.11 shows the starting situation of this scenario. This scenario demonstrates the effect of an unequal task load.

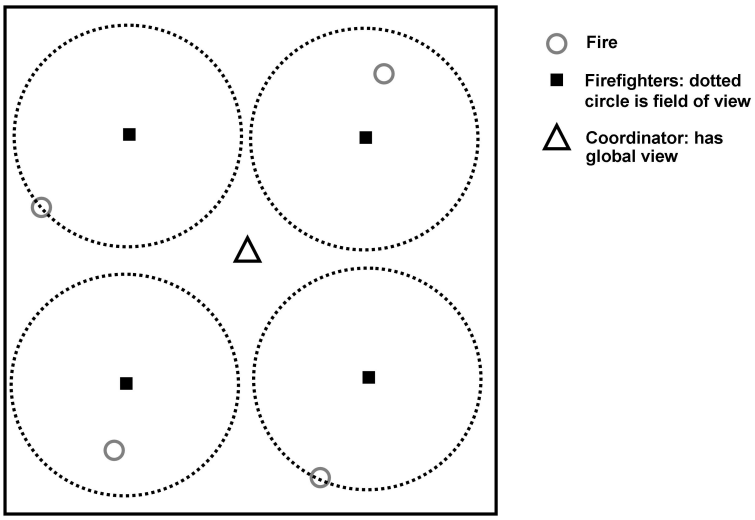


Figure 3.9: *Screen shot of the experimental environment: begin situation of the standard scenario A*

3.5.3 Results and Discussion

Table 3.4 shows the results of the simulations. The score is calculated by the time it takes until all fires have been extinguished. It is measured per scenario and coordination type. Scenario A shows no significant difference in the performance of both organizations. This is our standard scenario that shows that both coordination mechanisms work. In scenario B the firefighters reach a better performance based on their local information than the coordinator based on global information. The coordinator has no complete knowledge, and therefore he misses important information for his planning task and the cooperative firefighters stay inactive without commands. The time delay caused by the *invisible* fires is 7 seconds. In scenario C the fires are not equally distributed. Some of the firefighters cannot see the fires, which makes them inactive in the emergent organization. The global information of the coordinator shows to be more useful than the local information of the firefighters, because the coordinator's commands send the firefighters to the fires.

The results are as we expected. The difference between the two coordination mechanisms is that the decisions are made at a different level in the organization and based on different information. Both perform well in specific situations, none of them proved to be sufficient for all situations. The results suggest that in a scenario with a dynamic environment in which the agents experience these situations successively, both coordination types perform badly because of the weak points that are pointed out in the previous scenarios. In that case the best

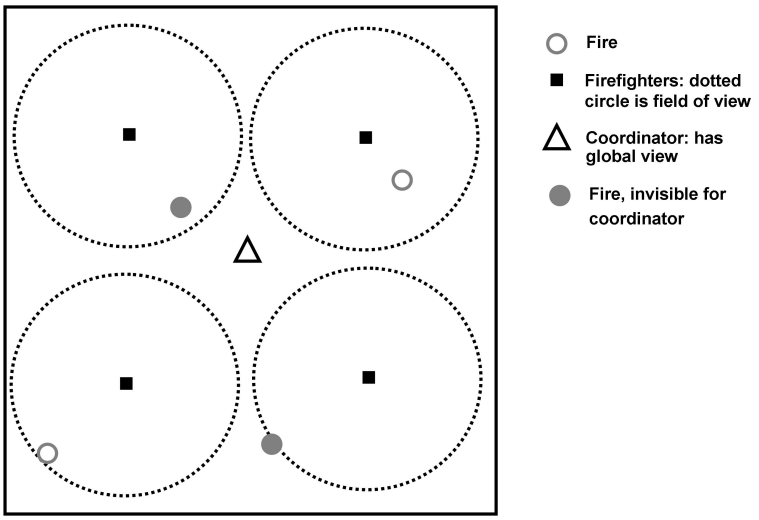


Figure 3.10: *Scenario B: No complete information for coordinator*

Table 3.4: *Results of our Experiment; time (s) until all fires are extinguished per coordination type and scenario.*

	Scenario A	Scenario B	Scenario C
Explicit Coordination: Self-reliant/Cooperative Agents	17.0 s	23.9 s	25.2 s
Emergent coordination: Non-Social Agents	17.2 s	16.4 s	35.0 s

organization would be one that dynamically switches between the coordination mechanisms.

3.5.4 Dynamic Coordination

Within the *explicit* and *emergent* coordination mechanisms the attitude of the actors with respect to the organization is fixed. We can achieve dynamic coordination by allowing the agents to make local decisions about their attitude. We want them to act following organizational rules, but also allow them to take initiative in specific situations. This is a form of *adjustable autonomy*. We believe that organizations in complex environments can benefit from agents that show *adjustable autonomy*.

We have created a third coordination mechanism, *dynamic coordination*, which combines emergent and explicit coordination. The dynamics are based on ad-

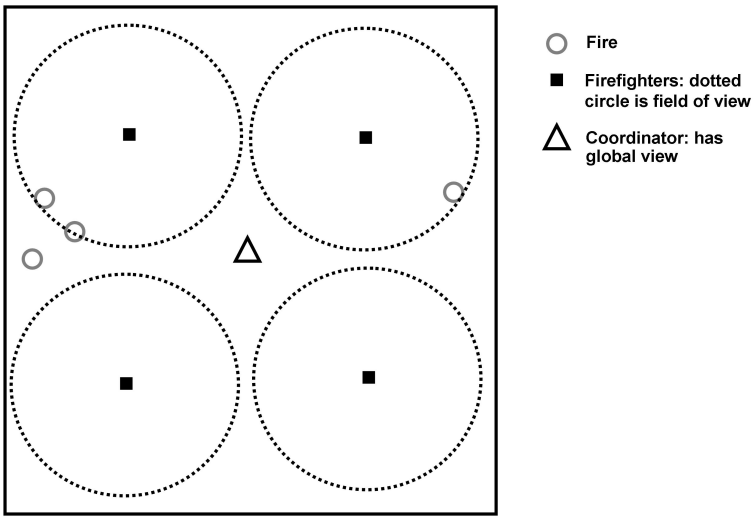


Figure 3.11: *Scenario C: no equal distribution of fires*

justable autonomy; the firefighter agents decide locally whether to follow the organizational commands or take initiative by themselves. The heuristic they use is as follows:

- if they receive an order from the coordinator, they will follow the order.
- if there are no orders and they observe a fire, they act pro-actively and extinguish they fire.

The intuition for those rules is that the firefighters are aware of the organizational goal to extinguish the fires as fast as possible. The firefighters actively try to contribute to the organization, which implies that they take the initiative to extinguish fires. On the other hand, they are cooperative to the orders of the coordinator.

We have run all three scenarios for the dynamic coordination mechanism. The results are given in Table 3.5. If we compare the performance of dynamic coordination with explicit and emergent coordination, we see that it performs the same as the best of the other coordination mechanisms in all scenarios.

The results from Table 3.5 suggest that a dynamic coordination mechanism can outperform the presented organizations in a dynamic environment. We have constructed a fourth scenario D, in which the situations from A, B and C occur after each other. When all fires of one scenario have been extinguished, the next scenario starts. As we can see in Table 3.6, the dynamic coordination mechanism outperforms the other two. Therewith, this experiment shows that dynamic coor-

Table 3.5: *Extended experiment; time (s) until all fires are extinguished per coordination type and scenario.*

	Scenario A	Scenario B	Scenario C
Explicit Coordination: Self-reliant/Cooperative Agents	17.0 s	23.9 s	25.2 s
Emergent coordination: Non-Social Agents	17.2 s	16.4 s	35.0 s
Dynamic Coordination: Adjustable Autonomy	17.0 s	16.4 s	23.0 s

Table 3.6: *Extended experiment; time (s) until all fires are extinguished per coordination type in a dynamic environment.*

	Scenario D: Dynamic Environment
Explicit Coordination: Self-reliant/Cooperative Agents	62.9 s
Emergent coordination: Non-Social Agents	64.6
Dynamic Coordination: Adjustable Autonomy	56.5 s

dination can be useful in dynamic environments, since it adapts its coordination type to the existing situation.

We want to point out that the aim of this experiment was not to construct the best coordination mechanism for our problem. We wanted to show that there is not one static coordination mechanism that is the best in all situations. Therefore, dynamic coordination seems to be useful in complex, dynamic environments.

We have demonstrated that dynamic coordination can be achieved in a *bottom-up* manner; each of the agents continuously determines the form of coordination of which it thinks it is best. The agents do so by controling external influences; they choose which knowledge they use for their decision-making process. They choose whether to use local observations, or to follow organizational commands. Therefore, the experiment shows that organizations can benefit from adjustable autonomy from the individual actors.

3.6 Towards Adjustable Autonomy

An autonomous agent has control over external influences. We have showed how we define attitudes based on the processing of external events. The modular

approach in our reasoning model makes an explicit division between influence control and decision making. Therefore, it allows the agent to reason about how to process external events.

In Section 3.2 we proposed a reasoning model which gives the agent a module to process external events explicitly. We have presented examples of event-processing rules that use knowledge from the belief base and result in an effect on the mental state. In Section 3.3 we have developed attitudes for agents by defining how they process external events. The experiment from Section 3.4 showed the effect of attitudes of the actors for organizational performance. Finally, in Section 3.5 we showed the need for a dynamic coordination mechanism in complex environments and we demonstrated that dynamic coordination can be achieved in a bottom-up way by allowing the agents to change their attitude. This is a form of adjustable autonomy.

3.6.1 Adjustable Autonomy

We have defined adjustable autonomy in Chapter 2 as: *dynamically dealing with external influences on the decision-making process based on internal motivations.* Adjustable autonomy is the process of reasoning about how an agent is influenced by external factors. An example of adjustable autonomy is an agent who reasons about how it observes the world. It can decide to focus on specific information based on its tasks, or choose to ignore certain sensors if it gets overloaded with information.

For another example of adjustable autonomy, we can think of situations where it makes sense to change the dependence with respect to other agents: if a fire-fighter is in danger he could ignore a request from the coordinator and give priority to his own goals in order to stay safe. He deliberately gives priority to his own goals, and therewith it takes the risk to violate the organizational norm.

In our reasoning model, adjustable autonomy is the process of reasoning about event-processing rules. An agent can have many event-processing rules. It can construct different attitudes at runtime. Sichman et al. [Sichman and Conte, 1998] and Dastani et al. [Dastani et al., 2003] describe several possible attitudes. The model presented in this paper allows those different types of agents. For example, it can decide to activate of deactivate event-processing rules, or it can change the prioritization order of the rules.

Situation-based prioritization is an example of meta-reasoning. This can be done by applying machine-learning techniques. A learning agent can learn the situations in which specific rules should have priority, such as the *danger*-example. Another option is to prioritize based on heuristics that can be defined in argumentation logics. Prioritization is studied in argumentation logics [Brewka, 1994]. Argumentation has been applied to reason about interaction between agents [Kakas et al., 2005] as well as to reason about organizational norms [Oren et al., 2008]. Our rule-based approach of event processing fits very well with this type of meta-reasoning. In Chapter 6 of this thesis, we discuss meta-reasoning further.

3.6.2 Heuristics for Event Processing

Adjustable autonomy is the process of reasoning about how external factors influence the reasoning process. In this section, we introduce some general heuristics an agent can use to process external events in an informal way. Later on, these heuristics can be subject of meta-reasoning.

One can think of several heuristics that can be used to process external events that together determine how and by whom the agent's mental state is being influenced. One of the heuristics based on which an agent can control external influences is *relevance of information*, as argued by Castelfranchi [Castelfranchi, 1995]. He defines relevance of information with respect to a certain goal. Intuitively, an agent might want to focus on a specific type of information given its goals. Reasons can be that it does not want to be distracted, or it wants to prevent information overload. Therefore it should be able to determine whether information is relevant for the goal. The reasoning rule we used as example in the previous section can be used here. The predicate *relevant* allows rules that enable the agent to focus.

In Chapter 2, we discussed the relation between autonomy and coordination. In order to achieve coordination agents need to influence each other's behavior. However, an agent might want to do one task autonomously, whereas with another task it allows input from others. In order to make the distinction the agent needs to know which information is relevant for which task.

Relevance of information is a heuristic for an agent to determine *how* it influenced. The agent should also to control *by whom* it is being influenced. The reasoning rules can use knowledge about the organization or about social context. Can the sender of a message be trusted? Does a request originate from a superior or from an unfamiliar source? These examples require social and organizational knowledge. Agents achieve coordination by allowing influence on the internal state based on social and organizational knowledge. Chapter 5 of this thesis describes how organizational rules can be translated into reasoning rules for event processing.

Other reasons to control by whom an agent is being influenced can be socially inspired, for example *trust*. Can the sender of a message be trusted? Trust can be defined as the acceptance of a new statement without further evidence. Several models for trust have proposed. [Barber and Kim, 2001] presents a model for internal belief revision based on the trust level of the other agents. The mechanism to determine the reputation of agents counts as the social context in the model. The use of trust in heuristics to process messages can be seen as social knowledge.

From the above we construct three different heuristics for influence processing:

1. Only accept information that is relevant with respect to current goals/tasks
2. Process messages/observations based on organizational norms
3. Accept messages only when the sender is trusted

The heuristics have a very general character and do not contain domain specific knowledge. They can be implemented in the component for event processing at the object-level. The knowledge and meta-knowledge that is used in the heuristics should be derived directly from the mental state of the agent. Therefore, the agent needs to be able to derive relevance of information. The social and organizational knowledge need to be present in the knowledge base. In the next chapters we specify these heuristics further. Chapter 4 focuses on information relevance; how to determine information relevance, what are the benefits of information relevance determination? In Chapter 5, we focus on organizational knowledge. We include organizational knowledge in event-processing rules. In Chapter 6, we discuss reasoning about the heuristics.

3.7 Conclusion

In this chapter, we included autonomy in a reasoning model. We motivated our choice for BDI-reasoning models, because their way of semantic reasoning is intuitive and it facilitates social interaction. We extended the BDI-reasoning model with a component for processing external events. We proposed the use of reasoning rules to specify how incoming events are handled. Those event-processing rules can be used to create attitudes of the agent with respect to the outside world.

Using an experiment we showed that different attitudes have an effect on organizational behavior. This shows the relation between autonomy and coordination. An extension of the experiment motivated the need for dynamic coordination mechanisms in complex environments. We demonstrated that this can be achieved in a bottom-up way by allowing the agents to change their attitude.

Finally, we presented general heuristics for influence control that can be used to process external events. Three general heuristics are based on information relevance, organizational knowledge and trust. Having control over external influences implies that the agent chooses when to use which heuristics. We look at this process as adjustable autonomy.

Chapter 4

Magic Agents: Using Information Relevance for Event Processing

4.1 Introduction

Agents are believed to be autonomous, meaning that they have control over external influences, as argued in Chapter 2. In a multi-agent environment where coordination of group behavior is required agents will influence each other. We have argued that an autonomous agent should control how and by whom it is being influenced.

Imagine a courier who takes goods from one place to another. He is receiving many delivery requests, but he can handle only some of them. He will have to make his own local decisions on which ones he will accept based on several considerations; he might have to follow contracts with suppliers, or he might prefer short distances above longer routes. This example illustrates different aspects that the courier should take into account when it processes incoming requests. Personal benefits as well as social or organizational motivations should play a role in the decision.

The agents in a multi-agent system are autonomous and make their own decisions. Nevertheless, in order to achieve coordination agents need to influence each other's decision making. Research on the relation between coordination and autonomy presented in Chapter 2 and Chapter 3 lead to a reasoning model for agents that guarantees autonomy and at the same time allows coordination. The model gives the agent explicit control over how it is being influenced. The agent uses local knowledge to process incoming events.

Key issue is to find general heuristics to control external influences. In this chapter we investigate *information relevance* as such a heuristic. An agent that can determine the relevance of information with respect to its goals, is able to dynamically deal with external input and is less sensitive for information overload

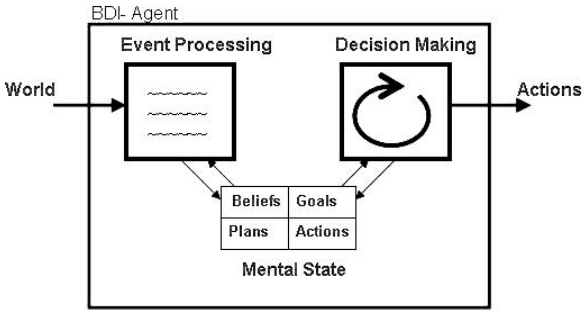


Figure 4.1: *Reasoning model of a BDI-agent including control over external influences.*

[Castelfranchi, 1995]. The agent should evaluate the relevance of new information with respect to its current plans and goals. We show the potential benefits of this heuristic in a computational model.

The process to determine whether incoming information is relevant depends on the reasoning model of the agent and how it stores its knowledge. If all the agent's knowledge are facts and the reasoning process uses only facts, information relevance can be calculated by looking at which knowledge is used in the decision-making rules. However, if the agent's beliefs are stored in deduction rules that derive more complex knowledge, relevance determination requires an analysis of the derivation process as well. Logical deduction is commonly used in agent knowledge-bases, for example in BDI agent model such as 3APL [Dastani et al., 2004b].

In this Chapter we describe a way to determine information relevance in BDI agents based on *magic sets* [Beeri and Ramakrishnan, 1991]. Magic sets have been developed for efficient deductive database searching. We introduce a new use of magic-sets theory that is beneficial for agent reasoning.

4.2 The Reasoning Model

In Chapter 3 an extension to the classic reasoning model has been proposed, such that an agent analyzes incoming stimuli based on internal knowledge, before it adopts them in its mental state. This requires a separate process next to the goal-directed decision making. Figure 4.1 shows the component for influence control. The agent deliberately chooses to adopt or reject external events based on its internal state. As we argued, the mechanism of having control over external influences is a requirement for agent autonomy.

The model uses reasoning rules in order to decide on adoption or rejection of certain influences. The reasoning rules contain predicates that consult knowledge from the agent's internal state. The agent determines the predicate values based

on local information. Reasoning rules have the following structure:

```
<head> <- <guard> | <body>
```

where the head indicates the activation event for the rule, the guard contains predicates that should match with the agent's belief base, and the body describes the resulting internal action, which is the effect of the event on the agent's internal state. An example of a reasoning rule for event processing is:

```
observe(X) <-  relevant(X) | UpdateBeliefs(X)
```

The rule is activated by an *observation* event. The head, guard and body state that if something is observed and it is relevant the agent will add the information to its belief base. An agent that has this rule only will process relevant observations and ignore other irrelevant observations. Another example of event processing can be that an agent rejects new observations as long as it is busy with a certain task:

```
observe(X) <-  busy() | IgnoreEvent()
```

These are examples of reasoning rules to control external influences. The agent evaluates the guard using its local knowledge, and therefore it actively chooses whether it should reject or adopt an event. This approach is opposed to models that adopt observations or messages directly into the belief base, where the external events have the control over the agent's internal state.

The agent processes external influences based on local knowledge. It evaluates predicates in reasoning rules to determine whether to adopt or reject influences. An immediate question is: what are valid reasons to adopt or reject influences? Here we can make the connection to coordination. The reasoning rules for event processing describe how and by whom an agent can be influenced.

The event-processing rules are separated from the decision-making process. Because of this modularity they can be changed and adapted easily. It allows an agent to change its attitude towards the environment at runtime by changing the event-processing rules.

4.2.1 Heuristics for Influence Control

One of the heuristics based on which an agent can control external influences is *relevance of information*, as argued by Castelfranchi [Castelfranchi, 1995]. He defines relevance of information with respect to a certain goal. Intuitively, an agent might want to focus on a specific type of information given its goals. Reasons can be that it does not want to be distracted, or it wants to prevent information overload. Therefore it should be able to determine whether information is relevant for the goal. The reasoning rule we used as example in the previous section can be used here. The predicate *relevant* allows rules that enable the agent to focus.

In Chapter 1, we mentioned the relation between autonomy and coordination. In order to achieve coordination agents need to influence each other's behavior. However, an agent might want to do one task autonomously, whereas with another task it allows input from others. In order to make the distinction the agent needs to know which information is relevant for which task.

Relevance of information is a heuristic for an agent to determine *how* it is influenced. The agent should also to control *by whom* it is being influenced. Therefore, the agent can use knowledge about the organization or about social context in the event-processing rules. Can the sender of a message be trusted? Does a request originate from a superior or from an unfamiliar source? These examples require social and organizational knowledge. Agents achieve coordination by allowing influence on the internal state based on social and organizational knowledge.

One can think of several other reasons to allow or disallow influence on the internal state in a certain context. Domain knowledge can always play a role. In this chapter we focus on the general heuristic of *information relevance*. In the next section we describe the use of this predicate further. The next chapter, Chapter 5, describes how organizational knowledge can be translated into reasoning rules for event processing.

4.3 Using Information Relevance for Influence Control

We have argued that an autonomous agent should control *how* and *by whom* its decisions can be influenced. We will use the notion of information relevance to specify the *how* part. In this section we discuss the benefits of information relevance determination. We present scenarios that intuitively show the use of information relevance and we explore the benefits of *information relevance* in a computational model.

4.3.1 Benefits of Information Relevance

An agent is continuously receptive for input from the environment, for example by its own observations or by messages from other agents. In our model an agent has the option to adopt or reject incoming stimuli based on certain considerations. In this chapter we focus on *information relevance* as such heuristic. We discuss the relation between information relevance and autonomy and coordination and we discuss relevance in the context of individual performance. We show how an agent can improve its performance by determining information relevance using a computational model.

Autonomy

Consider the courier agent from Section 4.1 that takes a package from A to B. Meanwhile it continuously receives requests to pick up some goods and take them

somewhere else. The agent can stick to its original plan and just deliver the package. It could also reconsider its plan only with extra options that concern goods that need to be delivered on the way from A to B, because the agent will profit from it and it won't be much extra effort. Third option is to replan the original plan based on all requests, even those that require a route from A to C to B.

If the agent simply decides its actions based on the presence of requests, it can still control the different options by allowing or rejecting influence on its mental state. Given that an agent can determine the relevance of a request with respect to the different planning options, it can choose one specific strategy by ignoring irrelevant requests for that strategy.

This is a way to control the autonomy level of decisions. An agent that takes its decisions purely based on its own observations, makes its decisions fully autonomously. If the agent allows input from other agents into its reasoning, the decision making is less autonomous. When an agent has the ability to adopt or reject external influences, it can control the level of autonomy of its decision making. This example shows how information relevance provides a heuristic to control the autonomy level.

Performance

Here we explain how controling external influences based on information relevance can improve the performance of an agent. Consider the following situation. An agent travels from A to B and wants to stay updated about the traffic information on its way. If there is heavy traffic, it should reconsider its route. The agent continuously receives messages with traffic reports. However, only the messages that concern information about the route between A and B are relevant for the agent. An agent adopting all traffic information will take more time to determine whether it should reconsider its route than an agent that only adopts the relevant traffic reports.

The general explanation is that an agent can make its decisions much faster if it only reasons with relevant information. A lot of irrelevant information may affect the quality of decisions in a negative way. If irrelevant information can be filtered out before the agent starts reasoning with it, the agent can perform better.

Information Overload

Knowledge about relevance of information can be used to filter on input for the decision-making process. It gives the possibility to protect an agent for information overload. We will give a practical example from real world robots. Consider a robot that is connected with a reasoning module. The connection between the robot and the reasoning module is such that observations of the robot produce events that need to be processed by the reasoning engine.

Many reasoning models are cyclic; they process the events once in every deliberation cycle. This makes that, during one cycle, the events are queued. As events can occur in a high frequency, for example by distance measurements, the queue can get too big to guarantee real time reasoning of the events.

This issue can be solved by hard coded rules that block certain event types. This is, however, a static and predefined way of filtering events. A better solution would be to filter the incoming events dynamically based on its relevance for the reasoning process. For example, when the robot is in a speech dialogue with somebody, sound events are highly relevant, whereas distance observations might not influence the reasoning process in any way. At such a moment you want to block the distance events.

4.3.2 Relevance as Filter

In this research, we introduce a method to filter events on relevance with respect to the reasoning process. By looking at the current goals and the active reasoning rules, we can evaluate which observations are relevant to the agent. In this section, we go into the process of using relevance determination as a concept in the reasoning model. We describe a BDI-agent that works with the reasoning model presented in Sect. 4.2.

The agent has a BDI-based decision-making process, which implies that the agent reasons in a goal-directed manner and it uses its beliefs in order to decide on its actions. The decision-making phase is preceded by a process for influence control as shown in Fig. 4.1. This phase contains reasoning rules that transform an incoming event into an effect on the agent's internal state. The reasoning rules have a *head-guard-body*-structure, where:

- Head: the event triggering the rule
- Guard: situational constraints that are checked with the belief base
- Body: the action causing the effect on the agent's internal state

Decision making on action is a process where the agent reasons with its goals and beliefs. The agent is continuously receptive to events that change it's beliefs and goals. The reasoning rules for event processing are a filter on the events, such that the agent controls which events influence its internal state. The heuristics that we mentioned in Sect. 4.2.1 are used in the guard of the reasoning rule.

In order to use information relevance as a concept to reason, when dealing with external events, we need to create a predicate $relevant(X)$ that describes the relevance of information X. We can also define relevance with respect to a certain goal: $relevant(X, G)$, where X is a piece of information and G is a specific goal. Using these predicates, we can create event-processing rules.

The set of event-processing rules that an agent has, specifies the way the agent perceives its environment. Let us stay at the example of information relevance.

An agent with the following rules allows only relevant observations to influence its decision making:

```
observe(X) <-  relevant(X) | UpdateBeliefs(X)
observe(X) <-  NOT relevant(X) | IgnoreEvent()
```

Here, the predicate describes relevance with respect to all the agent's goals and tasks. We can create as well a relevance predicate to describe relevance with respect to a goal, which is needed if the agent wants to focus on a task. We defined the predicate $relevant(X, G)$, where the first argument contains the new information, and the second argument can contains a specific goal. With the following rules, the agent can take the attitude, where it perceives only messages with relevant content for that goal:

```
message(X) <-  relevant(X, G) | UpdateBeliefs(X)
message(X) <-  NOT relevant(X, G) | IgnoreEvent()
```

The guard of the event-processing rule describes the situational constraints of the rule. More complex guards can be specified that distinguish between states of the agent. For example, if an agent only wants to filter its observations on relevance when its busy, we can create the following set of rules:

```
observe(X) <-  NOT busy() | UpdateBeliefs(X)
observe(X) <-  busy() AND relevant(X) | UpdateBeliefs(X)
observe(X) <-  busy() AND NOT relevant(X) | IgnoreEvent()
```

The event-processing rules specify the way the agent is influenced by external factors. The rules can be seen as a filter on the input. The guards contain the heuristics to filter the input, and information relevance is one of them. Next, we discuss the benefits of this heuristic in a computational model.

4.3.3 A Computational Model

We introduced *relevance* as a predicate for the reasoning rules to control external influences. The event-processing rules of an agent specify the attitude of the agent towards the environment. Earlier in Sect 4.3.1, we described the benefits of relevance determination by describing scenarios. The given examples support the idea to check incoming information on *information relevance* before adding it to the belief base.

Intuitively, we can say that relevance determination gets more useful when the decision-making process makes use of complex deductions, such that an increasing knowledge base leads to larger decision-making costs. Furthermore, the benefits of filtering information based on relevance grow as the percentage of irrelevant data gets larger.

However, determining the relevance of information of course also has its costs and therefore it might not always be beneficial. Therefore, it is required that relevance determination needs to be done quickly. Now, we illustrate the benefits of determining information relevance in a computational model.

Linear Complexity

Consider an agent that has to perform a task and for which it needs to derive information from its belief base. The time it takes to fulfill the task is dependent on the derivation time of the information, and therefore it is dependent on the number of elements in its belief base. The costs to perform task T are:

$$costs(T) = c_1 \cdot size \tag{4.1}$$

where c_1 is a constant value and $size$ indicates the size of the belief base.

The agent collects information continuously, for example by receiving messages. The agent receives N messages per time step. The costs to perform task T increase with the time, since the belief base increases. We can describe the costs as a function of time:

$$costs(T, t) = c_1 \cdot N \cdot t \tag{4.2}$$

Suppose that only part of the messages contains relevant information for the agent's task. An agent that does not care about information relevance adopts all incoming information and the costs for performing the tasks are as in 4.2. Another agent is able to determine the relevance of the information with respect to the task. The costs to determine the relevance of information is given by the constant c_2 for every piece of information. Given that he will only accept relevant information into its belief base, the costs function over time for this agent is:

$$costs(T, t) = c_2 \cdot N + c_1 \cdot R \cdot N \cdot t \tag{4.3}$$

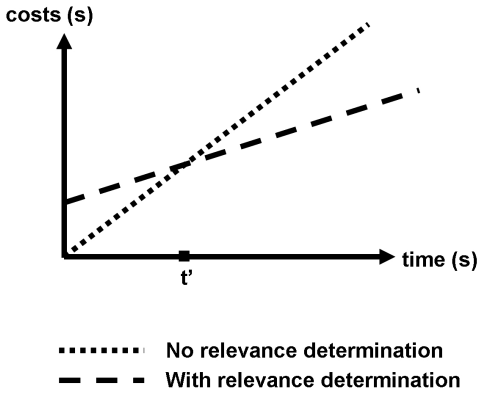
where N is the number of messages the agent receives per time step and R is the percentage of relevant messages, $0 < R < 1$. Graph (a) in Figure 4.2 shows a plot of both cost functions 4.2 and 4.3.

We see that the agent benefits from the determination of information relevance only after a certain $t = t'$. We can do some calculations to find out what this moment t' is. And given the previous cost functions, the following holds

$$t' = \frac{c_2}{(c_1 \cdot (1 - R))} \tag{4.4}$$

The moment where the agent performs better with relevance determination is dependent on the ratio c_2/c_1 and the percentage of relevant messages. In graph (b) of figure 4.2 we have plotted t' as a function of R. What we see is that if the percentage of relevant messages increases t' increases. When R approaches the value 1, t' becomes infinite. That means that if all information is relevant, there is no reason to check on relevance before adopting it. Furthermore, we see that the minimum value of time t' is equal to c_2/c_1 and this is the case when (almost) all messages are irrelevant. Also we can conclude that the number of messages an agent receives per second is not relevant for t'.

Given our cost functions 4.2 and 4.3, based on linear complexity equations in relation to the size of the database, we see that an agent can benefit only from information relevance if:



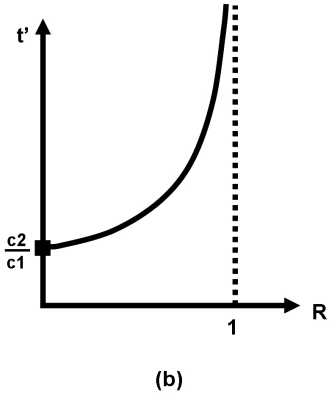


Figure 4.2: *Graph (a) Task costs with and without determining information relevance, and (b) t' as function of percentage relevant messages R.*

Table 4.1: *Beneficial moment t' for linear and non-linear cost function*

Ratio $c_2/c_1(s)$	Messages per second, N	factor relevant messages, R	$t'(s)$ linear costs	$t'(s)$ non-linear costs
100	1	0.5	200	11.5
100	100	0.5	200	1.2
100	1	0.01	101	10.0
10	1	0.5	20	3.7
10	100	0.5	20	0.4
10	1	0.01	10.1	3.2

- the number of irrelevant messages is reasonably high
- the benefit occurs only after a minimum time period determined by the ratio c_2/c_1

Non-linear Complexity

Often it is the case that information derivation from the belief base is not a process of linear complexity, but non-linear. This is for example the case when certain facts need to be deducted and recursion occurs. Below we change the cost function for the task into a non-linear function. Consider that the costs of information retrieval from the belief base is a function of the squared size of the belief base.

$$costs = c_1 \cdot size^2 \tag{4.5}$$

We have to rewrite function 4.2 and 4.3 into 4.6 and 4.7 for the costs as a function of time, where 4.6 does not determine relevance of information and 4.7 does.

$$costs(T,t) = c_1 \cdot (N \cdot t)^2 \tag{4.6}$$

$$costs(T,t) = c_2 \cdot N + c_1 \cdot (R \cdot N \cdot t)^2 \tag{4.7}$$

The cost functions are now parabolic. The moment t' where relevance determination shows better performance for the agent can now be calculated by

$$t' = \sqrt{\frac{c_2}{c_1 \cdot N \cdot (1 - R^2)}} \tag{4.8}$$

In this function t' depends on the ratio c_2/c_1 and R, but t' has also become dependent of the amount of information per time step, indicated by N. If the number of messages per second increases with a factor 100, t' becomes 10 times smaller. Table 4.1 compares the t' moments for the linear and non-linear case. We have taken a ratio of $c_2/c_1 = 100$ and $c_2/c_1 = 10$. The values of N and R are varied as well.

The general conclusion of the computational analysis is that relevance determination gets more useful when the decision-making process makes use of complex

deductions, where the deduction time depends on the size of the data base. Furthermore, the benefits of filtering information based on relevance grow as the percentage of irrelevant data gets larger. It supports the requirement that the relevance determination needs to be done quickly.

This computational model only uses examples of linear and quadratic complexity of decision making with respect to knowledge base size. Logical deductions in agent-based systems, as well as real-world decision-making can be more complex. In those cases, one benefits even more from the awareness of information relevance.

4.4 Relevance Determination

Agents can improve performance when they can determine the relevance of information at the moment of perception. This is especially the case in situations where agents collect a lot of data over time and the percentage of irrelevant data is high. Relevance, however, is a subjective concept. What is relevant according to an agent depends on its present knowledge. Here we discuss how to define relevance and we propose a method for relevance determination.

4.4.1 Information Relevance in AI

Although the benefits of information relevance are intuitive, there is not much to be found on information relevance and BDI-agents in related literature. Castelfranchi has described information relevance for BDI-agents [Castelfranchi, 1995], but he did not include methods to determine relevance of information.

The concept of *relevance* has been discussed in other research areas. In epistemology it is accepted that relevance is a subjective concept that depends on someone's current knowledge. The same holds for cognitive science [Sperber and Wilson, 1986]. In social interaction, for example, a communicative act automatically implies its relevance according to the actor, as the actor has chosen for it. If it was not relevant to the actor, he would not perform the action. To the hearer, however, the relevance depends upon the state of local knowledge when the communicative act is perceived.

As relevance depends on the currently available knowledge, a method for relevance determination depends on the reasoning model of the agent and its model for knowledge representation.

In probabilistic reasoning, several computational methods for relevance determination have been developed [Neal, 1996]. Information relevance is refered to as *information gain*. The information gain with respect to a conclusion is defined in terms of the changes it produces on estimations of the probability of the conclusion. Information gain has a quantative measure. Van Diggelen et al. use information gain for efficient information distribution is sensor systems [van Diggelen et al., 2008]. This approach is not applicable to logical reasoning as

it is commonly used in BDI-agents where reasoning based on semantic relations between objects instead of probabilities.

Logical deduction is commonly used in agent knowledge-bases, for example, in 3APL [Dastani et al., 2004b]. Knowledge is stored in terms of derivation rules and reasoning engines, such as Prolog, are applied. Here, relevance determination requires an analysis of the derivation process.

This reasoning process comes close to deductive databases [Elmasri and Navathe, 1994], where knowledge is stored in deduction rules. In the research area of deductive database, relevance has been a topic of several studies. The *magic sets* theory is a formal method that uses information relevance for efficient query processing in deductive databases.

In the magic sets approach, a fact is considered relevant if it might, depending on the database, be essential to derive an answer to a given query. This means that the method used for query processing, e.g. Prolog, defines the relevance of information.

In deductive databases, *magic sets* are used to increase the efficiency of query evaluation [Beeri and Ramakrishnan, 1991]. Given a query, the method evaluates the relevant data in the database. This is the opposite use of information relevance compared to our needs. The agents need to assess the relevance of a new piece of information that is observed. The main difference is that we focus on changing information, whereas in databases the information is static and the queries are dynamic. However, the logical features of *magic set* theory are comparable to the reasoning process of BDI agents. We can apply its view on information relevance to agent reasoning, and use magic sets for relevance determination.

4.4.2 Magic Sets

The reasoning process of an agent can be seen as asking queries to its belief base. Therefore, in order to study information relevance in the context of agent reasoning, we have investigated the concept of relevance in database research. In particular deductive databases have similarities with logic-based reasoning-models. A well known technique in deductive databases that makes use of information relevance is *magic sets*. Here we briefly explain the theory behind *magic sets*.

The magic set method is a bottom-up query evaluation technique first introduced in [Bancilhon et al., 1986]. A straight forward algorithm for the magic-set transformation is explained in [Beeri and Ramakrishnan, 1991]. Magic sets are used to define the relevant predicates for a specific query and are used to speed up the search process significantly.

Consider the program P, describing *route* relations based on *connected*-facts. Furthermore, we have a database with the *connected* relations, describing the map as shown in Fig. 4.3 and a query Q requesting the set of routes from position *a*.

```
P:
r1    route(X,Y) :- connected(X,Y)
r2    route(X,Y) :- connected(X,Z), route(Z,Y)
```

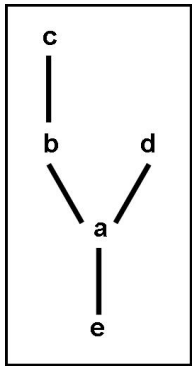


Figure 4.3: *The road map described by database D*

```
D:   connected(a,b), connected(b,c), connected(a,d), connected(e,a).

Q:    route(a,Y)?
```

There are several approaches to solve the query. Two standard strategies are the top-down strategy and the bottom-up strategy. Both have their strong and weak points. The magic-sets method tries to combine the two approaches. Below we describe the evaluation process using the top-down, bottom-up and magic-set strategy.

Top-down Search

Top-down evaluation is used by Prolog. It starts from the query, creates new subgoals from the deduction rules. When possible it tries to match the query variables with the facts in the database.

As example we use program P describing routes in a map. In the evaluation of query Q rule *r1* leads to a subgoal *connected(a, Y)*. Next, the facts *connected(a, b)* and *connected(a, d)* provide possible bindings for variable Y, namely $Y = b$ and $Y = d$, which leads to two possible answers to the query: $route(a, b)$ and $route(a, d)$.

Rule *r2* results in binding variable Z with the constants b and d and the new subgoals $route(b, Y)$ and $route(d, Y)$ are created. Repeating the process leads to the binding of variable Y with c. The answers to the query will be:

```
route(a,b).
route(a,d).
route(a,c).
```

Top-down evaluation works well for the example shown here. However, the efficiency is sensitive to the order of the right-side arguments of the deduction rule.

For example, if the arguments in $r2$ are switched, a Prolog search for all answers would run forever. Furthermore, a wrong variable binding will be discovered only after trying all optional subgoals. This happens in our program when a query such as $route(X, a)$? is solved.

Bottom-up Search

Bottom-up evaluation starts from the available facts. It derives new facts using the derivation rules. It computes all possible route relations and then matches the answers to the query. In the example given above, the bottom-up search strategy can immediately apply rule $r1$ and derive the following new facts:

```
route(a,b).
route(b,c).
route(a,d).
route(e,a).
```

In the second round the program can use the data from database D plus the newly derived facts. Now, rule $r2$ is applicable and the following facts can be derived:

```
route(a,c).
route(e,b).
route(e,d).
```

These facts are added to the database again. In a third round the program can derive another new fact:

```
route(e,c).
```

No more new facts can be derived. If we finally match the data to query Q, the result of the bottom-up evaluation is the same as the result of top-down:

```
route(a,b).
route(a,d).
route(a,c).
```

As can be seen in the bottom-up evaluation process also irrelevant route facts have been derived. The derived fact $route(e, a)$ is an example. This fact has not contributed to any of the final answers to the query. In total we have derived eight facts. As the route map from the example would have been bigger and it would contain e.g. routes from point e to another point f, the number of irrelevant facts would be even higher.

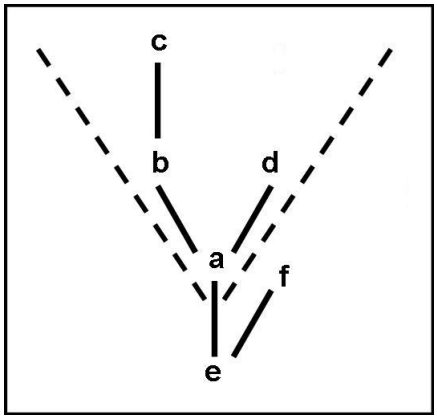


Figure 4.4: *The dashed cone depicts the relevant elements to solve the query* route(a, Y).

Magic Sets: Combining Bottom-up and Top-down

The magic-sets method extends bottom-up reasoning such that only relevant facts will be derived. If we go back to our example, we know intuitively that only routes starting from point a are relevant to the query $route(a, Y)$. This is visualized by Figure 4.4. The dashed cone in Fig. 4.4 represents the view on the database that contains the relevant points. In order to solve the query we need to compute route relations, but only those of the points that stay inside of the cone.

Informally, we can say that for any point i inside the cone i equals a or there is a route from a to i. Therefore, for any $route(X, Y)$ relation that contributes to our final answer, we know that the binding of the first variable X is a or that there is a route from a to that point. Using the magic sets-method we create a predicate *magic_route(X)* that possesses this property. This predicate is added to the derivation rules of *route(X,Y)*.

```
P':
r1'    route(X,Y) :- magic_route(X), connected(X,Y)
r2'    route(X,Y) :- magic_route(X), connected(X,Z), route(Z,Y)
```

Adding the magic predicate to the deduction rule of $route(X, Y)$ restricts the possible bindings of variable X. The magic predicate contains the relevant bindings of variable X, which we call the magic set. The notion of relevance comes from the top-down evaluation. Therefore, we can construct the magic set directly by applying top-down knowledge on the original program rules.

The magic sets transformation given in Appendix A describes how the magic predicate can be created from the derivation rules from program P and a top-down evaluation strategy. The definition of the predicate *magic_route* in our example is as follows:

```
r3'    magic_route(a).
```

```
r4'    magic_route(Z) :- magic_route(X), connected(X,Z)
```

The rules guarantee that any point in the magic set is a or there is a route from a to that point.

If we apply bottom-up evaluation to program P' and the routemap from our example as shown in Fig. 4.3, the first evaluation round leads to two new *route*-facts from rule $r1'$ and two magic facts from rule $r4'$:

```
route(a,b).
route(a,d).
magic_route(b).
magic_route(d).
```

Adding these facts to the database leads in the second round to derivation of two new facts:

```
route(b,c).
magic_route(c).
```

In the third round rule $r2'$ can derive the fact $route(a, c)$. After that, no more new facts can be derived. Matching the results to the query gives the following answers:

```
route(a,b).
route(a,d).
route(a,c).
```

Comparing the bottom up evaluation of the magic program P' with the original program P, the answers are exactly the same. However, no longer we have derived irrelevant route facts. We had to derive some extra magic predicates, but the total number of derived facts, 7, is one less than the original program. And expanding the roadmap with extra point outside of the cone would not lead to derivation of more facts, which would be the case in the original program.

Magic sets combine bottom-up evaluation with knowledge from top-down reasoning. The evaluation process is done bottom-up, but a constraint is added such that only relevant predicates are derived. This notion of relevance comes from top-down reasoning.

Magic Set Transformation

A general algorithm to apply the magic set transformation on any deductive database is given in [Beeri and Ramakrishnan, 1991] and are described in Appendix A. The algorithm to create a magic set for a program P and query Q contains the following steps. First, the adorned program P^{ad} needs to be constructed from a top-down evaluation method. Secondly, the magic predicates and their derivation rules are created. Last, the adorned derivation rules in P^{ad} are modified by adding by the addition of magic predicates to their body. The

achieved program Pm is equivalent to P with respect to the query. Any evaluation strategy applied to the magic program will restrict all predicate derivations to those that are relevant according to the top-down strategy. A more detailed description can be found in Appendix A.

4.4.3 Using Magic Sets to Define Relevance

The example from the previous paragraph shows how the magic sets approach increases efficiency in bottom-up evaluation. At this point, we are not interested in the most efficient search algorithm. The important part for our work is the definition of relevance. The example shows the construction of a magic set that contains all relevant variable bindings for the predicate for which it has been created.

We are interested in *information relevance* in the context of BDI agents. There seems to be a good match between our needs and the properties of magic sets. There is one difference between the way magic sets are used for bottom-up evaluation and our needs: the standard magic sets transformation considers derivable predicates only, whereas we are interested in the relevance of observable facts.

Consider the following program, where the predicate $route(X, Y)$ specifies all possible routes. Furthermore, it defines a predicate $on\_route(X)$ indicating that there is a route from the current position to point X. Also it describes how to detect traffic in rule $r3$; if is receives a traffic message about a location and for this location holds $on\_route(X)$, the message contains relevant information for traffic determination.

```
P:
r1    route(X,Y) :- connected(X,Y)
r2    route(X,Y) :- connected(X,Z), route(Z,Y)

r3    on_route(X) :- position(Y), route(Y,X).
r4    traffic() :- on_route(X), message(X).
```

Assume that the messages are observable facts. In our work we would like to know which messages are relevant for the query $traffic()$. Magic sets transformation on this program only creates magic sets for the *route* predicates that occur in the body of $r2$ and $r3$. However, there is no reason why we cannot create a magic predicate for the observable facts. It does not contribute to more efficiency in bottom up evaluation, but it provides us with relevance information about those facts. Therefore, we extend the magic sets algorithm such that we create magic sets for all predicates in the body of the derivation rules, not only for the derivable predicates. We can create a magic set for the messages, resulting in the following rule:

```
magic_message(X) :- on_route(X).
```

The above rule contains all relevant messages for the query $traffic()$ in program P, and therefore we can say:

```
relevant( message(Z), traffic() ) :- magic_message(Z).
```

We defined the relevance of messages in terms of its magic set. The relevance predicate can now be used in event-processing rules as shown in Section 4.3.2.

4.5 Magic Agents

We introduced *relevance determination* as a heuristic in reasoning rules to control external influences. From Sect. 4.3 we can conclude that a quick assessment on information relevance is important. Furthermore, we have discussed *magic sets* as a method to determine information relevance with respect to a certain query. Next, we introduce *magic agents* as agents that are able to determine information relevance using magic sets.

4.5.1 Information Relevance in BDI-agents

In work on agent autonomy Castelfranchi has described information relevance for agents with respect to goals [Castelfranchi, 1995]. According to his definition, information is relevant for a goal if the information is about the goal, if it is about the content of a goal or if it is about plan relations of the goal. Castelfranchi did not include any methods of how relevance of information could be determined. In order to do so, we abstract from the goals at first and look at relevance with respect to the agent's reasoning process in general.

We analyze the notion of relevance based on the BDI reasoning model *3APL*. 3APL, presented in [Dastani et al., 2004b] and [Hindriks et al., 1999], provides a reasoning model and a programming language for BDI-agents. The agent's internal state consists of a belief base, a goal base, a set of reasoning rules and a set of capabilities. The deliberation cycle describes the decision-making process, and makes use of the concepts in the internal state. Transition rules describe the deliberation process is a formal way.

During the deliberation process queries will be asked to the belief base. We say that information is relevant at the moment it is perceived, if the information contributes to at least one of the possible queries given the internal state of the agent at that moment.

From the formal semantics of the 3APL model we know that the following types of queries exist in the deliberation process: *guard checks*, *test actions* and *goal achievement checks*. The guard checks are used to activate a reasoning rule, test actions are tests on the belief base as part of a plan and goal achievement checks are used to check whether current goals have been reached.

In order to determine whether new information is relevant we need to consider the agent's internal state at the moment of perception. We know the set of possible queries given the internal state of the agent, as they are called by the reasoning rules. The reasoning rules have the form:

```
<head> <- <guard> | <body>
```

where the head should match the goals of the agent, the guard should match the agent's belief base and the body contains a plan to reach the goal.

As information is relevant with respect to an agent's goals [Castelfranchi, 1995], the goals constrain the set of reasoning rules that can be activated. Only the reasoning rules of which the head matches one of the goals need to be considered to determine the possible queries. The guards of those reasoning rules contain the relevant *guard checks*, and the body contains the relevant *test actions*. To complete the set of all possible queries we have to add the goals from the goal base to the set of queries since they stand for the goal achievement checks. The set of possible queries given the agent's internal state are:

- *guard checks* in reasoning rules matching the goals
- *test actions* in reasoning rules matching the goals
- *goal achievement checks* represented by the goals

4.5.2 Relevance Determination in 3APL

We describe the 3APL-reasoning as queries on a database. Based on the reasoning rules of an agent, we know all possible queries. All information used to solve those queries is relevant for the agent. However, the belief base also contains allows deduction rules and it will be difficult to tell in advance which facts are used to solve the query. We could evaluate all possible queries, in order to determine whether information is relevant at the moment of perception. This would be an intensive task. More important, we would not save any effort compared with just accepting all events, because the same query solving will be done in the reasoning process.

We can also try to describe relevance of information for each type of observation. The algorithm we developed is based on *magic sets*. We can create magic sets for the observation types, which automatically links the data to the relevant queries. When the agent receives new information, it can assess the magic sets for the relevance determination at runtime.

Every query is solved by a subset of data from the database as shown in Fig. 4.5. This set is the relevant set of data. An observation is relevant for a query if it is part of the relevant dataset of the query. The magic sets construct the view on the database that contains the relevant dataset.

4.5.3 Applying Magic Sets in BDI-agents

We need to determine the relevance of new piece of information, whereas magic sets define a set of relevant elements within a database. We can use the magic sets to determine information relevance in a BDI agent. If we consider the mental state of an agent as a database and the reasoning process as queries on the

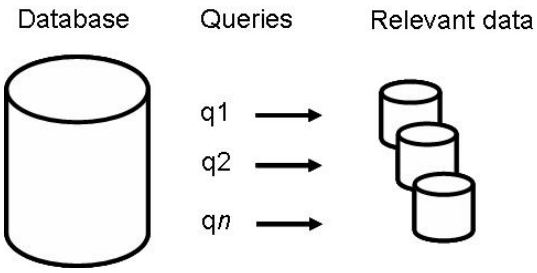


Figure 4.5: *Every query has a view on the database that contains all relevant data for the query.*

database, we know the possible queries at a certain moment as we showed in Section 4.5.1. We can create magic sets for those queries. The agent wants to know the relevance of an observation with respect to a query. Therefore, we create a predicate $relevant(X)$, which is derived using the magic sets of the queries, where the variable X holds an observation. We give an example of the use of Magic Sets in an agent setting.

Consider an agent traveling from X to Y. It continuously receives traffic information, and it continuously reconsiders its route based on the traffic information. New information might lead to a new route. Intuitively we know that only information that concerns the agent's possible routes is relevant. Information about other parts of the map will not influence the planning. We will construct the predicate $relevant(traffic\_message(X))$.

We have programmed the agent in 3APL [Dastani et al., 2004b]. The belief base contains the knowledge about all possible routes in the *route* predicate. Furthermore, the agent can determine whether a location is on an optional route. The reasoning rules contain the decision to start replanning the route from X to Y based on traffic information and otherwise execute the action *Go*.

```
BELIEF BASE:
  route(X,Y) :- connected(X,Y)
  route(X,Y) :- connected(X,Z), route(Z,Y)

  on_route(X) :- position(Y), route(Y,X).
  traffic() :- on_route(X), message(X).

REASONING RULES:
  go() <- traffic() | ReconsiderRoute
  go() <- NOT traffic() | Go
```

We apply magic-set transformation on the belief base. The standard magic-set transformation rewrites the predicates that occur in derivation rules. The algorithm leads to the following rules:

```
  route(X,Y) :- magic_route(X), connected(X,Y)
```

```
route(X,Y) :- magic_route(X), connected(X,Z), route(Z,Y)

magic_route(X) :- position(X).
magic_route(X) :- magic_route(Y), connected(Y,X)

on_route(X) :- position(Y), route(Y,X).
traffic() :- on_route(X), message(X).
```

The reasoning rules of the agent relate possible queries to goals. For every query that is used in the reasoning rules, we create a magic set that restricts the relevance of the query to the goals of the agent. In this case, the query $traffic()$ occurs in the reasoning rules for the goal $go()$. Therefore, we define the predicate $magic_traffic()$ and use $goal(go())$ as seed for the magic traffic set:

```
magic_traffic() :- goal( go() ).
```

As stated, we need to magic sets for the observable elements, in this case the definition of $magic\_message(X)$ predicate. Therefore, we apply the magic set transformation to create the magic set belonging to the fact $message(X)$:

```
magic_message(X) :- magic_traffic(), on_route(X).
```

The set of relevant variable bindings of $message(X)$ is in the magic set, which feeds the *relevance* predicate. The deduction of $magic_message(X)$ tests on $magic_traffic()$, which finds out of the query $traffic()$ is relevant according to the current goals. The predicate $on\_route(X)$ defines whether the message is actually used for the query $traffic()$.

Finally, we define the relevance predicate. We want to determine the relevance of traffic messages. Therefore, we need to create a magic set for the *message* predicate. We define relevance of a traffic message as follows:

```
relevant( message(X) ) :- magic_message(X).
```

All the above rules add up to relevance program R, which is added to the belief base. Whenever the agent receives a message it can check with this program whether the message is relevant or not according to its current state.

```
R:
  route(X,Y) :- magic_route(X), connected(X,Y)
  route(X,Y) :- magic_route(X), connected(X,Z), route(Z,Y)

  on_route(X) :- position(Y), route(Y,X).
  traffic() :- on_route(X), message(X).

  magic_route(X) :- position(X).
  magic_route(X) :- magic_route(Y), connected(Y,X)

  magic_traffic() :- goal( go() ).
  magic_message(X) :- magic_traffic(), on_route(X).

  relevant( message(X) ) :- magic_message(X).
```

4.5.4 An Algorithm for Relevance Determination

In this section we describe an algorithm to construct the relevance predicate that the agent can use in event-processing rules. The relevance predicate is part of the general knowledge of the agent; it can be stored as derivation rules in the agent's belief base. The predicate itself can be used in event-processing rules to control the agent's autonomy, section 4.3.

We describe the reasoning process as queries on a database. Based on the reasoning rules of an agent, we know all possible queries given the goals of the agent. Every query is solved by a subset of data from the database. This set is the relevant set of data. An observation is relevant for a query if it is part of the relevant dataset of the query. We propose a strong definition of relevance:

Definition. An observation is relevant for a query if is used to find a solution for the query.

The definition implies that the information is used to find a solution to the query. This definition comes close to the actual meaning of magic sets. Therefore, we use magic predicates to define the relevant data set of a query. The derivation rules for this magic predicate are constructed in the magic-sets transformation algorithm. The magic set transformation is taken from [Beeri and Ramakrishnan, 1991] and described in Appendix A.

We assume that we know the type of observations of which we want to determine the relevance. The algorithm to define the set of relevant observables for a query using magic sets can be described in the following steps:

1. apply the magic sets transformation to the deduction rules in the belief base.
2. create magic sets for the queries in the reasoning rules. The seed for the magic set is the goal that corresponds to the reasoning rule.
3. create magic sets for observable elements (for details, see step 2 from the magic set transformation in appendix A).
4. for each observation type, define a relevance predicate of the form

```
relevant( observation( Var ) ) :- magic_observation(Var)
```

All the above rules add up to a *relevance program*, which is added to the belief base. The relevance program specifies whether new information is used in queries for the current goals, given the current beliefs of the agent. Now, the agent has created knowledge about the relevance of information. This relevance knowledge can be used in rules for event processing.

4.6 Implementation of Magic Agents

So far, we have explained the theory behind magic sets in section 4.4.2 and in section 4.5 we have presented an algorithm to apply magic sets transformation

on BDI agents. The result is an algorithm to construct the magic predicates that enable determination of information relevance. In this section, we discuss the implementation of magic agents. When should the magic sets be created and how should they be evaluated? Furthermore, we present an experiment where we compare agents with and without the ability to process events based on relevance.

4.6.1 Creating Magic Sets

The algorithm for magic-sets transformation makes use of the syntax of the existing derivation rules. From the syntax magic predicates and their derivation rules are defined. The actual facts in the belief base of the agent are not needed for the algorithm. This gives the choice to apply the magic sets transformation at development time or let the agent perform the algorithm at runtime.

When it is clear at development time which observation types will be subject of relevance determination, the magic predicates can be constructed immediately. Then, the assumption that the reasoning rules of the agent are fixed. If the reasoning rules change the derivation rules of the magic predicates change as well.

It is imaginable that an agent's reasoning rules change at runtime, for example with agents that have the ability to learn. This can have implications for the derivation rules of the magic predicates. Or it could be that for a new type of observation, new magic predicates need to be constructed. The algorithm presented in 4.5 can be applied anytime, so there is no restriction. For the examples we use further in this section, we just assume that the magic predicates have been defined.

4.6.2 Evaluation of Relevance Predicate

There is a second choice in testing whether new information is actually part of the magic set or not. Magic sets have been developed to increase the efficiency of bottom-up evaluation, by using knowledge from top-down reasoning to rewrite the derivation rules. The theory shows that the magic set contains the relevant data for evaluating a certain query, given the current available data. Therefore, we use the magic set to determine information relevance. There are several ways to implement the relevance-determination process.

There is a choice in testing to see whether new information is actually part of the magic set or not. The evaluation of the magic predicates can be done once to determine all relevance elements. For new information, the agent needs to check whether it is in the solution set. Consider relevance program R, as constructed in section 4.5. The relevance predicate and the magic predicate are given by:

```
magic_message(X) :- magic_traffic(), on_route(X).
relevant( message(X) ) :- magic_message(X).
```

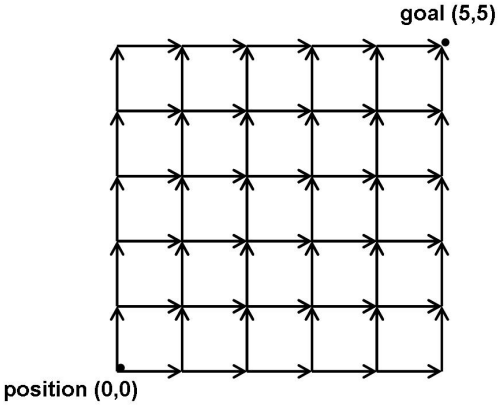


Figure 4.6: *Road map of a 5x5 grid*

When the agent determines the relevant set at once, it can evaluate *magic_message(X)*. Evaluation of the query *magic_message(X)* leads to all possible answers of variable X. The answers contain all locations that are used to determine the query $traffic()$. The agent can store all possible anwers in its memory and check incoming information on the answer set.

However, if the beliefs or goals of the agent change, the relevant set of elements might change as well, and the magic set should be evaluated again. The relevant messages are dependent of the goal $go()$. If this goal is inactive, the magic set will be empty. Then if the goal $go()$ is adopted the magic set changes. The magic set is also dependent on the position of the agent; if the position of the agent changes the relevant messages change. The same holds for the route map that is stored in the agent's belief base. Therefore, when beliefs and goals of the agent change often, it is not useful to evaluate all relevant messages via the magic predicate.

The other option is to evaluate the magic predicate at the moment new information arrives. The agent just checks whether a specific message is inside the magic set. For example, if the agent receives a message about location a, it evaluates *relevant(message(a))*, and therewith *magic_message(a)*. Evaluation of this query takes the current beliefs and goals into account.

4.6.3 Experiment

We have implemented the agent from section 4.5 and placed the agent on grid representing a route map, as shown in picture 4.6. The arrows indicate directed connections between two locations. The starting position of the agent is $(0, 0)$, and it's goal location is the upper right corner.

The agent receives traffic messages about a location of the form *traffic_ message(X)*. The location can be part of the route map of the agent, or not. In the

Field	Magic set evaluation	
	# Solutions	Time (s)
9x9	1	0.00
100x100	1	0.157
200x200	1	1.357

Figure 4.7: *Evaluation time of all relevant messages, via query* magic_message(X)

Field	Magic set evaluation	
	# Solutions	Time (s)
5x5	252	0.032
7x7	3432	0.312
9x9	48620	5.973

Figure 4.8: *Evaluation time of one relevant message*

case it is on the road map, the message contains relevant information for the agent in order to plan its route. In Figure 4.6, location $(1, 5)$ is an example of a relevant location. If the location is not on the road map the message is irrelevant for the agent's current task, for example a traffic message about $(6, 6)$.

First, we evaluate the calculation time of the complete set of relevant messages via the query *magic_message(X)* while we increase the grid size. Table 4.7 shows the results. As can be seen, the number of solutions to the query increases rapidly, and therewith, the calculation time. When the grid gets bigger, evaluating all possible relevance messages is no longer an option in real time.

The time of relevance determination of a specific location was evaluated via the query *relevant((X,Y))*. As shown in Table 4.8, the evaluation via magic sets is quite fast. We increased the field size up to 200. The evaluation time is the average over all possible locations. We conclude that the agent can better evaluate the relevance of a specific message when it arrives, than create the complete set of all relevant messages once in this particular example.

Next, we have implemented the agent in two models. One agent was implemented following the classical 3APL approach, which means that incoming messages are immediately stored in the belief base. This agent has no control

over external influences; its decision making-process is immediately influenced by the incoming events. Its belief base increases over time as more messages are received.

We have implemented the second agent with the reasoning component to control external influences. The agent accepts only relevant messages, and ignores all other messages. The following event-processing rules are used:

```
message(X) <- relevant(message(X)) | UpdateBeliefs(X)
message(X) <- NOT relevant(message(X)) | IgnoreEvent()
```

The agent has the ability to determine relevance based on magic sets. Its belief base is extended with relevance program R from section 4.5.

In a simulated environment, they received a number traffic messages per second with relevance percentage 5. We increased the number of messages they received per second. We measured the performance of the agents by the time it took the agent to decide whether or not to reconsider its route. Performance analysis showed that after time the agent with relevance determination was much faster in its decisions. The time gained by a smaller belief base and less belief updates was bigger than the effort of determining information relevance. The table shows similar results compared with the computational approach. The results show that filtering observations on relevance can improve the decision-making process of the agent.

4.7 Reasoning about Information Relevance

We have argued that an agent can control its autonomy via reasoning rules using information relevance. Relevance can be seen as meta-knowledge. It can be used in event-processing rules. In the examples we have shown, the agents determined relevance with respect to its goals and tasks. But there are other options to use relevance. We can construct relevance predicates as well for specific queries.

Consider a courier agent traveling from A to B using a road map as in the Figure. Meanwhile it receives requests to pick up and deliver goods. The agent could also reconsider its plan only with extra options that concern goods that need to be delivered on the way from A to B, because it will profit from it and it won't be much effort. Second option is to replan the original plan based on all requests, even those that require a route from A to C to B. These two possibilities can be distinguished in the agent's belief base via different derivation rules. We create two predicates:

```
easy_option(X) :- message(X), go(Y, Z), on_route(Y, X, Z).
hard_option(X) :- message(X), go(Y, Z), possible_route(Y, X, Z).
```

In our magic agent we now need define different magic predicates for messages, in order to distinguish relevance with respect to the specific query. The result looks the as follows:

```
relevant( message(X), easy_option ) :- magic_message1(X)
relevant( message(X), hard_option ) :- magic_request2(X)
```

Both relevant rules relate to different queries. In reasoning rules to control influences the agent processes events based on relevance in combination with other motivations. For example:

```
request(Y) <- low_priority_order AND
       relevant( (request(Y), hard_option() ) |
       Accept(Y)
request(Y) :- high_priority_order AND
       relevant( (request(Y), easy_option() ) |
       Accept(Y)
```

The agent determines which knowledge is used for decision making by using a filter on the input.

4.8 Discussion

We have presented a perspective on agent autonomy that can be modeled by event-processing rules to control external influences. Furthermore, we have shown a method to determine information relevance in BDI-agents, using the definition of relevance that matches with the magic sets approach as it is used in deductive databases and we have applied it to 3APL reasoning. In this section, we reflect on some of the assumptions we have made and we discuss some opportunities of our approach for the area of agent systems.

4.8.1 Alternative Relevance Definitions

We have taken a definition of information relevance, saying that information is relevant to the agent if it is used in the evaluation process of possible queries at this moment. The algorithm for the construction of the relevance predicate is based on the definition of relevance.

Alternative definitions lead to a different set of relevant data. Suppose we define relevance as follows: *An observation is relevant for a query if it can be useful for a solution of query, now, or in the future.* This definition would imply that any fact that is used in the derivation of the query is relevant, whether other predicates can be derived or not.

We have adopted our definition from deductive database theory, as we consider agent reasoning as query evaluation on the belief base. This view on agent reasoning fits with logic-based reasoning models such as 3APL. The logical features of those reasoning models allow for the adoption of formal theories from other research fields such as deductive databases.

4.8.2 Drawbacks of Filtering on Relevance

In our examples and the implementation we have demonstrated an agent that filters incoming information on relevance. We are aware that this is not always a good strategy, as information that you have rejected can become relevant later on. We kept the simple as our aim here was to demonstrate how information relevance can be determined and used in agent reasoning.

The agent evaluates relevance of information for one specific moment. In many cases it will not be beneficial to throw away whatever is irrelevant at the moment of perception. Since the environment changes or an agent's goals change, irrelevant information might become relevant. However, information overload is a problem in practice. A compromise could be to buffer irrelevant information instead of throwing it away. The irrelevant information does not affect the decision making since it is not in the belief base, and it could then be recovered when the situation changes. And we want to guarantee autonomy in agents concerning coordination. This research provides a reasoning model and tool to deal with it. We described a general way to determine information relevance without the use of domain knowledge.

Furthermore, information relevance should be seen as a heuristic to process incoming information. The agents filtered irrelevant information continuously, whereas, when discussing our model of autonomy, we stated that an autonomous agent should dynamically deal with external influences. Event then the agent needs to have knowledge of basic concepts such as information relevance presented in this chapter.

4.8.3 Opportunities of Using Information Relevance

We used information relevance to control external influences at runtime. The notion of information relevance in general and of magic sets specifically allows other interesting functionalities as well.

An agent could also be programmed to *clean up* the belief base at specific moments. After a period in which the agent has adopted a lot of new information, it might be useful to filter the belief base using information relevance now and then.

Another functionality that can be constructed using the properties of information relevance is that to structure the agent's belief base such that beliefs are linked to plans or goals for which they are relevant. This could speed up the decision-making process of the agent and could make the belief base more accessible. The agent should be aware of the relevance of information with respect to the tasks it performs.

4.9 Conclusion

In this Chapter, we have worked out the heuristic of information relevance. Potential benefits of using information relevance for influence control can be found in controling the decision quality and preventing information overload. The benefits of relevance determination get more significant when the decision-making process makes use of complex deductions, where the deduction time depends on the size of the data base. Furthermore, the benefits of filtering information based on relevance grow as the percentage of irrelevant data gets larger.

We have proposed a way to determine the relevance of information in BDI-agents using the *Magic Sets*-method from deductive database theory. Using the magic set transformation algorithm, we construct the magic set of observations from the deduction rules in the belief base and the reasoning rules in the plan base of the agent. The agent can check the relevance of new observations based on the magic sets. We have implemented the process in 3APL agents.

Chapter 5

Influence Control and Organizational Rules

This chapter discusses how autonomous agents can adopt organizational rules into their reasoning process. Agents in an organization need to coordinate their actions in order to reach the organizational goals. Organizational models specify the desired behavior in terms of roles, relations, norms and interactions. We have developed a method to translate norms into event-processing rules of the agents. We propose a modular reasoning model that includes the organizational rules explicitly. Since the agents are autonomous, they will have their own reasoning rules next to the organizational rules. The modular approach allows for meta-reasoning about these rules. We show that this stimulates bottom-up dynamics in the organization.

5.1 Introduction

Organizations benefit from autonomous decisions by their participants. This is visible in human organizations. Not only the formal organizational structure but also the informal circuit of communication and interaction between actors determines the success of an organization. In human organizations a participant's contribution is evaluated based on the organizational requirements as well as the extra achievements. Someone who takes initiative and builds up a personal network is often higher valued than someone who sticks to the official rules and does not do anything extra. The capability to act and make decisions in unexpected situations is usually perceived as a positive characteristic of human actors.

How does the observation that organizations benefit from participants' initiatives translate to multi-agent coordination mechanisms that are based on organizational theory? Every organization is created for a specific objective. The organizational model describes the desired global behavior using abstract concepts

such as roles, relations and norms. Its specification is meant to guarantee certain requirements, for example, about the information flow. However, since the agent is assumed to be an autonomous entity, decision making is a local process of the agent. Therefore, it is important to maintain agent autonomy within the multi-agent coordination model. The organizational rules should guide the choices of the agent, but the organization cannot control the agent's decision-making process.

In this research, we investigate how to make agents aware of organizational rules. At the same time we allow them to take initiatives besides the formal structure. We propose a modular approach with which agents can adopt organizational rules into their reasoning model. The reasoning model separates the organizational rules from the actual decision-making process. This way, the agent's decision-making process can be defined separately from the coordination mechanism. At the same time, the modular approach allows for metareasoning about different behavioral rules, which makes the agent independent from the organizational structure. The agent is not limited in its decision making. It knows how to follow the organizational norms and it is able to take other initiatives. Therefore, the model guarantees agent autonomy.

The chapter is structured as follows. First we discuss related work on agent organizations. We motivate our choice to use the OperA model to describe organizations and we give an example of the use of OperA. Next, we describe a reasoning model with which an agent can adopt organizational constraints to its decision making and we show how the organizational rules are adopted by the agent. Then we discuss how the agent and the organization come together. We investigate bottom-up dynamics in organizations using the autonomy of agents and we give examples. Finally, we conclude the chapter.

5.2 Autonomy in Organizations

From an organizational perspective, the organizational rules influence the autonomy of the agent. One of the perspectives on autonomy discussed in Chapter 2, defines autonomy as the ability and permission of actions [Bradshaw et al., 2004, 2005]. In their coordination models, Bradshaw et al. define the permissions and obligations of agents. Therewith, they adjust the autonomy level of the agents. The more freedom the agent has in choosing its actions, the more autonomous it is. In other works on autonomy, there is a relation between autonomy and coordination as well. For example, the perspective of social autonomy [Carabelea et al., 2004; Falcone and Castelfranchi, 2001], deals with autonomy in relations between agents. In a hierarchical relation, the autonomy of an agent depends of the abstraction of delegated tasks.

In our research we assume that the agents are autonomous entities in the sense that they have control over external influences. This implies that the organizational model specifies behavioral guidelines for the agents to assure desired features such as task coordination or information flow. The agents should follow

those guidelines, but they are not forced to do so by definition.

Researchers in multi-agent systems have introduced the organizational metaphor to achieve coordination between autonomous agents. Organizational models specify coordination mechanisms between agents in abstract concepts, such as roles, relations and norms. We use human organizations as inspiration. From this point of view, we consider an organization as a description of roles, relations and interactions to achieve certain coordination. The agents fulfilling the roles are autonomous.

We discussed related work on autonomy and organizations in Chapter 2. We described how different approaches allow agents to take up organizational tasks and we described the consequences for the agents' autonomy. We briefly recall the conclusions here.

5.2.1 Organizational Models

Several models for agent organizations have been proposed. Ferber et al. introduced the Agent Group Role (AGR) model [Ferber and Gutknecht, 1998], which used the concepts of roles, groups and interaction between roles. The agents in the AGR model are autonomous; they are outside of the organization. However, the organizational rules specify the desired behavior in terms of actions and goals. A normative approach to describe behavioral guidelines would leave the agents more space in choosing how they actually pursue the goals that belong to their role.

The OperA model [Dignum, 2004] proposes a more expressive way for defining organizations by introducing an organizational model, a social model and an interaction model. This approach explicitly distinguishes between the organizational model and the agents that act in it. Agents become aware of the organizational rules via contracts that specify these rules. The agents are still fully autonomous in making decisions.

Another approach is Moise+ [Hübner et al., 2002], which also separates the organizational model from the agent model. In order to operationalize the organization, a middleware has been developed that checks whether actions of agents are allowed or not according to the governing organizational rules. The middleware can overrule the decisions of the agents. The organization becomes an active entity that has the possibility to interfere in the agents decisions. Therefore, the agents are not fully autonomous in executing their choices. The same holds for the electronic institutions [Esteva et al., 2001; Garca-Camino et al., 2007]. Electronic institutions are norm-based coordination models for open multi-agent systems. The norms are specified in a multi-agent middleware, to regulate agents' actions. The institution actively interacts with the agents.

Other organizational models, as introduced by Matson [Matson and DeLoach, 2005], are based on formal semantics and transition rules. The possible states of the organization are described by state transitions. The state description captures the whole system; they include the organizational structure as well as the internal

knowledge of the agents. Matson [Matson and DeLoach, 2005] includes internal knowledge of the agents within the specification. The agents are restricted by the organizational specification. According to several definitions, autonomous agents should have control over their internal state. Therefore, the autonomy requirement is not fully met in this case.

The OperA approach is compatible with our notion of autonomous agents and it has expressive semantics to define organizations. Therefore, we have chosen to use OperA for the organizational specification. In order to use OperA, we still need a mechanism to describe the adoption of organizational contracts into an agent's reasoning model. In Section 5.4 we show how autonomous agents can adopt organizational rules. In the following section we will give an example of an organization specified in the OperA model.

5.2.2 The OperA Model

OperA [Dignum, 2004] provides a formalized specification language for agent organizations. OperA describes an operational organization in three parts:

- The organizational model: roles, relations, interactions
- The social model: population of organization, linking agents to roles
- The interaction model: describes interactions given organizational model and agents

The organizational model contains the description of the roles, relations and interactions in the organization. It is constructed based on functional requirements of the organization. The social model and the interaction model are the link between the organizational description and the executing agents. Here the organizational rules are translated to contracts for the agents fulfilling the roles. OperA includes a formal language to describe those contracts.

In an operational organization the social model and the interaction model can be dynamic, because of agents entering or leaving the organization. The organizational model is in principal static as long as no structural changes are carried through. The administrative tasks to keep track of the different organizational models are specified in organizational roles.

Agents enacting roles in a organization are expected to have some minimal knowledge about the concepts that are used to set up social contracts. The contracts are described in deontic expressions. The agents need to know the deontic concepts *permission*, *obligation* and *prohibition*. Furthermore, the description includes relations between roles. The agent needs to know the meaning of such a relation. For example, a *hierarchical* relation between role *r1* and *r2* implies that a request from *r1* is interpreted as an obligation by *r2*. OperA presents a formal description of the relevant concepts.

5.3 Example of an Organizational Description

In this section we present an example of an organizational model in OperA. We use a fire brigade to illustrate how an organization is specified and how the behavior rules for the agents are constructed.

The fire brigade operates in a world where fires appear that need to be extinguished. In the organization we define two roles; *coordinator* and *firefighter*. The coordinator makes a global plan and tells the firefighters which fire they should extinguish. The coordinator has a global view of the world. The firefighters perform the actual tasks in the environment; they move to a fire location and extinguish the fires. They have only local views.

There is a hierarchical relation between the two roles, the coordinator is the superior of the firefighters and can send orders to the firefighters, which fire they have to extinguish. We want to show different forms of coordination within this organization. In our implementation, we achieve this by changing the autonomy level of the decision-making process of the firefighters.

A generic methodology to analyze a given domain and determine the type and structure of an application domain resulting in a OperA agent organization model is described in [Dignum et al., 2004a]. The methodology provides generic facilitation and interaction frameworks for agent societies that implement the functionality derived from the coordination model applicable to the problem domain. Standard organization types such as market, hierarchy and network, can be used as starting point for development and can be extended where needed and determine the basic norms and facilitation roles necessary for the society. A brief summary of the methodology is given in table 5.1.

Table 5.1: *Methodology for designing agent organizations*

	Step	Description	Result
OM	Coordination Level	Identifies organization's main characteristics: purpose, relation forms	Stakeholders, facilitation roles, coordination requirements
	Environment Level	Analysis of expected external behavior of system	Operational roles, use cases, normative requirements
	Behavior Level	Design of internal behavior of system	Role structure, interaction structure, norms, roles, scripts
SM	Population Level	Design of enactment negotiation protocols	Agent admission scripts, role enactment contracts
IM	Interaction Level	Design of interaction negotiation protocols	Scene script protocols, interaction contracts

We focus on the organizational model or our firefighter organization. Below we define the coordination level, environment level and behavior level.

5.3.1 Coordination Level

At the coordination level, the coordination type of the society is determined based on the characteristics of the problem domain. There are several possibilities, for example a hierarchical model, a market based model or a network model. A hierarchical model is the most common structure in crisis management organizations such as our group of firefighters. The following characteristics are typical for a hierarchical organization:

- The leading goals for agents are global, organizational goals
- Relations between agents are fixed
- Communication is specified by design

We have chosen for a hierarchical organization. Based on the choice for a hierarchical model we define the environment level and behavior level.

5.3.2 Environmental Level

In the environment level, interaction between the organization and the environment is analyzed. Ontologies are needed to define organizational concepts and to define communication. Furthermore, the functional requirements of the organization are specified. This includes the global organizational purpose and the local objectives of the roles. We define coordination rules in terms of norms.

Organizational function.

The purpose of our firefighter organization is to detect fires in the environment and extinguish them as soon as possible.

Ontologies.

Besides OperA concepts to specify the organizational model, we need a communication language between the agents. We will use four performatives for the communication between the agents: *Request, Accept, Reject, Inform.*

Secondly we need a domain-level ontology to describe all objects in the environment, and the actions that the agents can communicate and reason about. Our domain ontology consists of one object, *Fire*, and one action, *Extinguish* and three states that describe the status of an agent with respect to its tasks: *Busy, Done, Free*

Roles.

We do not consider external stakeholders of our organization. We will describe the roles in our organization. The roles are based on a functional analysis of the task of the organizational purpose. The roles are described in Table 5.2.

A hierarchical organization needs a root role to take care of delegation of roles. The root role will give the role definitions to the agent highest in hierarchy and provide it with a social contract to specify the required behavior. In our case, the highest role is the coordinator role.

The coordinator role has as objective to hire firefighters. It will assign the firefighter rule to applicant agents.

Furthermore it has the objectives to monitor the fires in the world, to monitor the firefighters, and to assign fires to firefighters. For the last two objectives the coordinator is dependent on the firefighters. We specify the firefighter role with the objectives to extinguish fires, to inform the coordinator about its status.

Table 5.2: *Role table of the firefighter organization*

Role	Relation to Society	Role Objectives	Role Dependencies
Applicant	Potential member	Join organization	Root
Root	From hierarchy model	Assign coordinator role to applicant	Applicant
Coordinator	From hierarchy model	Monitor fires	
		Assign firefighter role	Applicant
		Monitor status of firefighters	Firefighter
		Assign fires to firefighters	Firefighters
Firefighter	Realization of extinguishing fires	Extinguish fires	
		Inform about status	Coordinator

Dependencies between the roles appear from the description of the role objectives. We define the coordinator as the highest role in the organization. Therefore it has a hierarchical relation with the firefighter role, where the coordinator is the superior of the firefighters.

Norms.

We specify the norms that hold between the coordinator and the firefighter roles in table 5.3. In the first norm we describe how a firefighter agent handles requests to extinguish fires. The firefighter is obliged to accept the request from the coordinator. This norm follows directly from the hierarchical relation between the two roles.

We define a second norm telling that the firefighter should keep the coordinator informed about its status. The information is needed by the coordinator in order to do its tasks properly. This norm guarantees the required information sharing within our organization.

Table 5.3: *The norms of the firefighter organization*

Norm	
Situation	Handling extinguish-request
Responsibilities	Initiator: coordinator Action: firefighter
Triggers	Pre: coordinator sends extinguish-request Post: coordinator is informed about accept
Norm Specification	**Whenever** extinguish-request from coordinator **then** firefighter **is** obliged **to do** accept-request

Norm	
Situation	Announce status
Responsibilities	Initiator: firefighter Action: firefighter
Triggers	Pre: status change Post: inform coordinator about status
Norm Specification	**Whenever** status-change **then** firefighter **is** obliged **to do** inform-coordinator-about-status

5.3.3 Behavior Level

Here, we describe the social model and the interaction model as defined in OperA. Typically, a hierarchical organization has a relatively detailed social model and interaction model. This implies that the norms in the social model and the communication protocols in the interaction model do not leave much space for individual contracts with the agents. We assume that the behavior rules as described in this level of the development of the organization match with the contracts with the agents.

Social Model.

In the social model we define the social contract for the agents that fulfill the roles. We describe the dependencies between the agents in more detail. The coordinator role just specifies the role objectives, with no further obligations. For the firefighter role we have defined some additional norms:

(1) Whenever $extinguish-requestfromcoordinator$ then $firefighter$ is *obliged* to do $accept-request$

(2) Whenever $status-change$ then $firefighter$ is *obliged* to do $inform-coordinator-about-status$

As we explained, the first norm follows directly from the hierarchical relation between the two roles. Therefore, the social contracts for firefighter agents should

only capture the second norm.

Interaction Model.

The interaction model describes interaction contracts. An interaction contract between two agents describes a protocol that is followed by the agents during interaction scenes. [Dignum et al., 2003] presents a formal language to specify the interaction between agents in behavior rules.

The agents have interaction contracts for all interaction scenes. In our example, there should be interaction scenes and contracts between applicant and root and between applicant and coordinator roles. However, they are only relevant to set up the organization. We will focus on the interaction contracts between the coordinator and firefighter roles here, as this interaction will occur during the execution.

Interaction contracts are agreed upon by agents playing the roles and encountering interaction scenes. We propose possible interaction contracts between agents playing firefighter and coordinator, Table 5.4.

Table 5.4: *The interaction contracts of the firefighter organization*

Interaction Contract	
Parties	Coordinator C, Firefighter F
Scene	Extinguish request
Clauses	**If** received(F, C, extinguish-request(fire)) **then** F **is** obliged **to do** answer(F, C, accept-refuse)

Interaction Contract	
Parties	Coordinator C, Firefighter F
Scene	Announce Status
Clauses	**If** received(C, F, status-report) **then** nothing

The first contract specifies the interaction between the firefighter and coordinator for the situation where the coordinator sends a request to the firefighter to extinguish a certain fire. The contract specifies that the firefighter is obliged to answer whether it accepts or rejects the request. The second contract specifies the interaction for the scenes in which the firefighter informs the coordinator about its status. The agents agreed that the coordinator does not need to respond.

The interaction contracts from Table 4 pose another behavior rule on the firefighter agent:

(3) Whenever extinguish-request from coordinator then firefighter is obliged to do answer-accept/refuse

5.3.4 Towards an Operational Organization

We have specified the organizational model in terms of roles, relations and interaction and we have defined contracts for agents that want to participate in the organization. Although agents are autonomous entities, we expect them to follow the organizational rules. When an agent joins an organization, it should adopt a contract that contains these rules.

We do not want to specify the internals of an agent when we specify the organization. We have developed a reasoning model that represents the organizational rules separately from the agent's decision-making rules. Therefore, the adoption of contracts can be done dynamically. In the next section we describe the agent's reasoning process in further detail.

5.4 Autonomous Agents in Organizations

The assumption that agents are autonomous entities is important in agent organizations. The organization specifies coordination at a high level. The agents should be aware of the organizational rules, but they still make their own decisions.

In Section 5.2, we have discussed other research concerning coordination of autonomous agents. The AGR model [Ferber and Gutknecht, 1998] gives a low level and detailed description of the desired behavior. The organizational middleware of Moise+ [Hübner et al., 2002] makes that the organization itself becomes an active entity, whereas we let the organization exist only in the agents. The organizational models based on formal transition rules [Matson and DeLoach, 2005] do not meet our requirement that the agent's internals have to be defined separately from the organization. We have adopted the OperA approach that describes the coordination via contracts for the agents joining the organization.

We believe that a modular approach in the agent's reasoning model is a promising way to adopt organizational rules into the decision-making process. By separating the organizational rules from the decision-making process of the agent we can guarantee the autonomy of the agent. Furthermore, the adoption of contracts can be done dynamically and the contracts can be changed at runtime.

In this section we show how autonomous agents can adopt organizational rules into their reasoning model. We explain briefly the reasoning model as described in Chapter 3.

5.4.1 Autonomous Decision Making

Being autonomous implies that agents have control over their internal state and over their behavior. The first implies that an agent determines its beliefs by itself. The second implies that decision making on action is a local process.

Coordination implies that agents influence each other in order to coordinate their behavior. Therefore, an agent allows influence on its mental state. For example, if an agent accepts a request to do a certain task from another agent, it

adds the task to its goal base. We argue that an autonomous agent controls how other agents influence its internal state.

Our approach to operationalize autonomy is to give the agent a choice in processing external events. The agent makes its own decisions, and it decides how other agents can influence its decision-making process. We introduce a component in the agent's reasoning model to deal with external events. We show how organizational rules can be specified in this reasoning component.

5.4.2 Event Processing in the Reasoning Model

Here we summarize the reasoning model we described in Chapter 3 and we show how it can be used to adopt organizational rules into the reasoning process. In the agent's reasoning-process we distinguish a component for event processing and a phase for decision making. The event-processing component gives the agent control over how it is being influenced by external events. The decide component focuses on the decision of action.

In the event-processing component the agent prepares the decision-making phase. External influences are processed here. External influence can be an agent's observations or messages from other agents. We have chosen to implement event processing with reasoning rules of the same form:

```
<HEAD> <- <GUARD> | <BODY>
```

The head of a rule is the event that triggers the rule. The guard should match the beliefs of the agent. The body of the rule expresses the influence of the event on the agent's reasoning process. For example, the following message describes that a request of a superior is to be accepted:

```
message(SENDER, request, TASK) <- superior(SENDER)
  | AddGoal(TASK)
```

The message is the trigger of the rule and the guard verifies with the agent's belief base whether the sender is a superior of the agent. If so, the request is accepted by adding the task to the agent's goals.

5.4.3 From Organizational Rules to Event-Processing Rules

We use the rules for event processing to specify how an agent's internal state is influenced by external events. The organizational specification describes behavior rules that are meant to guide the agent's decision-making process. We propose to translate the organizational rules from the organizational description to event-processing rules for agent decision-making.

We have to define the set of required elements to translate organizational rules to event-processing rules. Analysis of the OperA model shows that norms are based on triggers. The triggers, which can be messages or observations, are external events, and therefore can be used as head of the event-processing rules. In

the following, we propose to use a general message format consisting of a sender, a performative and the actual content. Observations must have the same format as agent beliefs.

The guards of the event-processing rules are restrictions based on internal beliefs of the agent. A guard can contain any set of beliefs. The example event-processing rule of the previous paragraph shows an example.

The body contains the effect of the external event on the internal state of the agent. This is of course dependent on how the internals of an agent are represented. If we consider BDI-agents, for example, the effect of external events then is described in terms of belief changes and goal changes of the agent.

5.4.4 Effects on the Mental State

Even when we consider BDI-agents, there are several ways to translate the organizational rules into internal state changes. One possibility is to transfer everything into beliefs that containing the result of the norm; *obliged(Action))*, *permitted(Action))*, etc. If this option is chosen, the assumption is that the decision-making process knows how to deal with those beliefs in such a way that the norms are fulfilled.

Another option is to translate the obligations directly to goals, and permissions and prohibitions into beliefs. The result of a event-processing rule would be internal actions to add goals or beliefs: *AddGoal(Action)*, *AddBelief(prohibited(Action))* or *AddBelief(permitted(Action))*. Then, an obligation leads automatically to a goal, and the fulfilling is in the hands of the deliberation cycle of the agent, where goals are selected. Prohibition and permissions are added to the belief base in order to make better decisions on action, without forcing anything.

It is possible to choose to remove mental elements. For example a prohibition may lead to remove the goal if it existed in the goal base. Drawback is that the goal cannot automatically be recovered if the situation allows it later on again.

In a more sophisticated set-up the decision-making procedures allow for different options concerning the effect on the mental state. For example the BDOING framework [Dignum et al., 2002] specifies an agent that derives its goals based on several aspects: norms, obligations, desires. In this framework, an obligation does not automatically lead to a goal, but can be stored as an (organizational) obligation.

If the agent has the capability to derive the deontic concepts based on derivation rules, one can add more specific constraints as well to the agent's belief base. For example, time constraints or contextual constraints can be added that describe the validity of an obligation, permission or prohibition over time.

The internal structure of the decision-making process determines the possible effects on the agent's mental state. This shows the generality of our approach. The consequence is, however, that we cannot present the one way of translating organizational rules that would work for all agent types.

For the remainder of this chapter, we propose to translate permissions and prohibitions with the *AddBelief* predicate to describe that the task is permitted or prohibited. Obligations will be added to the goal base directly using *AddGoal*.

5.4.5 Specification of Event-Processing Rules

We describe behavioral rules as event-processing rules that result in a change of the agent's mental state. Now we give an example of how to translate organizational rules into event-processing rules. In Table 5.5, we show possible values of the elements that we use to construct those rules. We assume that we can translate all organizational rules into event-processing rules by using those elements.

Table 5.5: *An ontology of event-processing rules that describe organizational rules*

Rule element	Description	Possible values
Head	External event that triggers the rule	- message(Sender, Performative, Content) - observation(Content)
Guard	Situational constraints	Any belief set of the agent
Body	Effect on agent's mental state	- AddGoal(Goal) - AddBelief(Belief) - IgnoreEvent()

5.4.6 Prior Organizational Knowledge

In Section 5.2.2 we explained that some minimal prior knowledge is required. An agent taking up a role in the organization should know the meaning of deontic concepts and of relational terms. The meaning of the deontic concepts *obligation*, *permission* and *prohibition* are part of the ontology for event-processing rules. We translate an obligation for the agent using the *AddGoal* predicate. This predicate adds the task directly to the goal base of the agent.

The agent should know the meaning of relational concepts, such as the hierarchical relation that we use in our example organization. The meaning of a hierarchical relation between an agent and its superior can be described by the following behavior rule: *Whenever an agent receives a request from a superior then the agent is obliged to accept the request.* When we translate this behavioral rule to an event-processing rule using the above described elements we get:

```
(r1) message(SENDER, request, TASK) <- superior(SENDER)
   | AddGoal(TASK)
```

We used this rule as example of an event-processing rule in Section 5.4.1. The above rule is considered to be general knowledge of the agent. The rule does not belong to a specific organization or a specific role.

5.5 Adopting Organizational Rules

Taking up a role in an organization means that an agent is expected to act following the constraints described in the role specification. In this section we show how organizational rules can be translated to event-processing rules for the agents. In the previous section we have shown how the required organizational prior knowledge is captured by the language specification and by some event-processing rules.

We continue with specific organizational rules using the example of the fire brigade. The firefighter organization shows three behavioral rules that a firefighter agent has to follow in that role. The rules are described by the norms in the social contracts and in the interaction contracts. The behavioral rules as described in Section 5.3.3 are:

1. Whenever *extinguish-request from coordinator* then *firefighter* is *obliged* to do *accept-request*
2. Whenever *status-change* then *firefighter* is *obliged* to do *inform-coordinator-about-status*
3. Whenever *extinguish-request from coordinator* then *firefighter* is *obliged* to do *answer-accept/refuse*

The first rule directly follows from the semantic meaning of the hierarchical relation between coordinator and firefighter. Therefore it is not part of the social contract. The other two rules are organization specific and need to be described explicitly. As discussed in Section 5.4.3 the rules are triggered by events. They hold when certain conditions are true, and they result in expected reaction of the agents. We can translate those rules directly to reasoning rules for event processing using the language elements presented in 5.4.3.

```
(r2) observe( status-change ) :- TRUE |
        addGoal( send(coordinator, inform, new-status) )

(r3) message( coordinator, request, extinguish(F) ) :- TRUE |
        addGoal( Send-answer(coordinator, request, extinguish(F)) )
```

Organizational rules are part of the social contracts and interaction contracts that the agent agrees upon when its joins an organization. They can directly be transferred to event-processing rules. These reasoning rules capture all behavioral rules that belong to the role which the agent has taken up. We show that adopting those reasoning rules is a way to make the agent aware of the organizational constraints.

The agent adds those reasoning rules to the event-processing phase that we have defined previously. This phase determines the degree of external influence an agent allows into its reasoning. The agent limits the autonomy level of the decision-making phase with the organizational constraints. Next we show that the agent can reason about those rules, and therefore it actively controls how it is being influenced.

5.5.1 The Event-Processing Rules

Earlier in this chapter we have described the firefighter organization and the organizational contracts that hold for the agents performing the firefighter role. Now we show how all parts come together and we show that the autonomy of the agents is guaranteed.

An agent adopting the firefighter role adopts three objectives: *extinguish fires*, *inform about status*, and *announce assistance*. This can be read from Table 5.2. If no further coordination is specified, the agent can pursue those objectives however it wants. In its decision-making process, the agent chooses the actions that lead to the objectives. But the agent has to deal with organizational rules.

In the reasoning model, we have added the event-processing phase that precedes the decision-making phase. We define event-processing rules that specify the effects of events on the agent's mental state, given certain conditions. Personal preferences of the agent can be described, for example that the agent ignores incoming requests when it is busy with other tasks. Also, general interaction rules are specified, for example, a rule that captures the meaning of a hierarchical relation.

If an agent joins an organization, it adopts knowledge about relations between roles. The general interaction rules together with the knowledge about role relations provide coordination. Furthermore, the agent adopts organizational norms by translating them to event processing-rules. Via this mechanism the agent allows the norms to restrict or guide its decision-making process. In Table 5.6 we have listed the reasoning rules from the example organization and we have added a personal preference rule $r4$.

Table 5.6: *The event-processing rule of a firefighter agent*

Rule Id	Origin	Rule
r1	General interaction rule	message(SENDER, request, TASK) :- superior(SENDER) | AddGoal(TASK)
r2	Firefighter organization	observe(status-change) :- TRUE | addGoal(send(coordinator, inform, new-status))
r3	Firefighter organization	message(coordinator, request, extinguish(F)) :- TRUE | addGoal(send-answer(C, request, extinguish(F)))
r4	Personal rule	message(SENDER, request, TASK) :- busy() | IgnoreEvent()

5.5.2 Modularity

In our reasoning model we have separated the event-processing rules from the actual decision-making process. This modularity has the advantage that it can

reason about those rules. The agent knows the origin of the event-processing rules. Because the agent can make this distinction, it has the possibility to prioritize the event-processing rules and therewith it deliberately chooses to follow specific norms.

When organizational rules are embedded in the actual decision-making process, the agent will follow the norms implicitly. It might not be aware anymore of which norms it follows and it might not be aware of which norm belongs to which organization or role.

Another advantage of the modularity is that it becomes easy to add or change the event-processing rules, and thus change organizational norms. This process would be more complicated when the organizational rules are mixed in the decision-making rules.

5.5.3 Guarantee of Autonomy

The agent adds the event-processing rules derived from the organizational contracts to own event-processing rules. We assume that, when an autonomous agent agrees with a contract, it deliberately chooses to do so. We further assume that the rules for event processing are possible and correct representations of organizational norms, so if the agent follows the event-processing rules it automatically follows the organizational norms.

The modularity in our approach allows for more complex reasoning about the event-processing rules. We claim that this guarantees autonomy of the agent; it knows how to follow the organizational rules, but it still has the possibility reason about them and take the chance to violate organizational norms. In the next section we give examples of metareasoning.

5.6 Dynamics in Agent Organizations

All static coordination mechanisms have their advantages and drawbacks. In a dynamic situation it is not possible to choose one coordination type that will always lead to the best performance. The main reason is that unexpected situations can occur that were not known at design time and that may not fare well with the selected coordination mechanism.

We have described a mechanism based on organizational concepts that specifies the coordination rules at an abstract level. At the same time, it preserves the actors' autonomy. However, the specified interaction rules in organizational models are static. There are two ways to achieve dynamics in an organizational model:

- Top-down: a new organizational model is defined, and the agents change their contracts with the organization. As a consequence they adopt different reasoning rules for influence control, which will change the coordination.

- Bottom-up: the agents change the priority of reasoning rules for influence control by themselves if they notice that the organizational model fails. They adjust their autonomy to repair the organizational failure.

The top-down dynamics can be achieved by carrying out structural changes, whereas bottom-up dynamics originate in autonomous choices of the agents.

5.6.1 Top-down Dynamics

The ability to reorganize is a powerful feature of an organization. Reorganization is a coordination problem by itself. Any reorganization process, however, only has an effect if the participants change their behavior. Therefore, we describe it both from an organizational perspective and an agent perspective.

The Organizational Perspective

Reorganization is a complex process, however. Dignum et al. [Dignum et al., 2004b] discuss issues on reorganization. There are several aspects of an organization that can be adapted. The writers make a distinction between behavioral changes and structural changes. Examples of behavioral change are new agents fulfilling the roles or new interaction patterns between agents. Structural changes affect the organizational model, such as the creation of new roles or behavior rules. Furthermore, the process of reorganization needs to be structured. The performance and effectiveness of the organization need to be monitored in order to make a decision on the moment and type of reorganization. Also, the adaptation itself needs to be controlled and communicated to the participants.

Hübner et al. present a practical implementation of a reorganization process using Moise+ [Hübner et al., 2004], where they specify roles with the objective to facilitate the reorganization process. They use the model to apply reorganization to switch between tactics in a soccer team. The structure of the soccer team as well as the coordination mechanism for reorganization are specified in Moise+. The agents in the organization can fulfill roles in both processes.

The Agent Perspective

Any reorganization process only has an effect if the participants change their behavior. Therefore, it puts requirements on the abilities of the agents participating in the organization. Models for reorganization specify the reorganization process, and assume that the agents adapt their behavior according to the organizational requirements. We claim that the modularity in the agent reasoning-model as presented in this thesis facilitates reorganization. The reasoning model as introduced in Chapter 3 distinguishes between influence control and decision making. The organizational rules are defined in the component for influence control, and the

agent can change those rules at runtime. That way, the agent processes adjustments of a organizational contract. The agent can deal with both behavioral and structural changes.

We describe a scenario, where the organization changes the coordination mechanism in a top-down manner. We start with the organization of the firefighter brigade that we have specified earlier in this chapter. The firefighter agents have adopted the organizational rules as event-processing rules as listed in Table 5.6. The rules specify a coordination mechanism in which the agents are obliged to inform the coordinator about a status change, and that they have to answer an *extinguish request* from the coordinator with an *accept* or *reject* message. The coordinator is the highest in the hierarchy. We assume that the coordinator role is extended such that the coordinator can take the initiative to change the coordination process.

If, for some reason, the current coordination mechanism causes too much communication over the network, the coordinator can take the initiative to change the interaction protocol. The agents can decide on a new protocol that leaves the reply of the firefighters out. The firefighter translates the new interaction contract to event-processing rules and ends up with the same set of rules, but without rule $r3$. The new interaction protocol has of course an effect on the information circulation within the organization, and therewith on the performance.

A more rigid example of reorganization can lead to a new role division. Imagine that the coordinator agent gets overloaded by too many tasks. It can assign the new coordinator role to a firefighter agent to take over the coordination in a certain area. Furthermore, he informs other firefighters that they get a new superior. The firefighter agent becoming coordinator adopts the behavior rules belonging to the new role. The firefighters who get a new superior update their beliefs according to the new situation. This example of reorganization has been implemented by Ghijsen et al., for an simulated ambulance organization [Ghijsen et al., 2008].

Martin and Barber [Martin and Barber, 2006] achieve dynamic coordination in group decisions. They propose several decision-making styles for groups ranging from command-driven to locally autonomous. In between the two extremes there are several ways to achieve consensus about a decision, for example via voting or a market-based approach. The best decision-making style depends on the situations. The researchers present a model where the agents switch from one decision-making style to the other. Our approach to connect the coordination mechanism with the reasoning-process of the individual agents matches well with their approach. The decision-making styles can be described in contracts using OperA. Each decision-making style is translated to event-processing rules that are implemented in the individual agents.

These examples show the effect of reorganization on the individual agents. The proposed reasoning model gives a way to process organizational changes at the individual level at runtime. Another issue with respect to agent autonomy is that it is possible to give the agents local control over the adaptation of their reasoning process. Therewith, they keep the control over their behavior. In that

case, the agents explicitly need to agree to organizational changes.

5.6.2 Bottom-up Dynamics

In Chapter 3 we showed dynamic coordination by allowing the agents choose their autonomy level. The agents acted following organizational rules, but also decided not to follow the rules in specific situations. This is an example of bottom-up dynamics. Organizations in complex environments can benefit from adjustable autonomy of agents. In this section we argue how metareasoning about event-processing rules can achieve bottom-up dynamics in the organization.

Furthermore, an organization can benefit from the pro-activeness of agents and from the 'informal' actions and communication. Every organization is created for an objective. Its specification is meant to guarantee certain requirements, for example, about the information flow. The agents are often able to do many things outside of organizational specifications without violating any organizational rule.

Both issues, metareasoning and informal processes, are directly related to the autonomy of the agents. We will discuss them in more detail and describe the dynamics in organizations using example scenarios.

Informal Processes

In the introduction we argued that participant's initiatives are another benefit for the organization that follows directly from agent autonomy. An organization is always specified for a certain purpose, possibly conflicting with the agents' individual purposes. Furthermore, the organizational rules guarantee required features, such as information flow, in order to optimize its performance. However, the agents are free to do what they want besides the organizational guidelines.

For example, interaction protocols are defined to guarantee a certain distribution of information, but the agents can chat with each other and exchange knowledge without violating the norms. In that case, the information distribution is larger than required by the norms only, which could make the organization more robust.

These informal processes are especially interesting when unexpected events occur that affect the organizational coordination mechanism. A scenario in the firefighter organization can demonstrate the benefits of informal communication between firefighters clearly. In our example, the organizational rules specify the communication between the coordinator and the firefighters. Nothing is said about mutual communication between the firefighters. This implies that it is not forbidden to communicate. Therefore, we consider communication between the firefighters as informal communication. If two firefighters share their knowledge about a fire while extinguishing it, the extinguishing process might go faster. As a consequence, the organization performs better due to the informal communication.

Unexpected events that undermine the coordination can be overcome by informal processes. If, for example, the communication between some of the firefighters

and the coordinator falls out, the organizational specification fails. The information flow as defined by the interaction protocols does not lead to the optimal knowledge for the coordinator; he misses the status of some of the firefighters. The informal communication between firefighters can be used by an individual firefighter to restore the information flow. If one firefighter tells another firefighter about its status, and this firefighter communicates it to the coordinator, the information flow in the organization is restored.

Metareasoning about Event-Processing Rules

An autonomous agent controls its internal state. Therefore, it should have control over external influences. The modular feature of our approach allows so. The event-processing rules are derived from a contract with an organization. They can be role-specific. At the same time an agent can have event-processing rules from other roles, or from itself.

We can tag the event-processing rules with their origin. For example, rule $r1$ from Table 5.6 captures the meaning of a hierarchical relation, and as such it is part of the agent's knowledge. It is not organization or role-specific. Rules $r2$ and $r3$ from Table 5.6 are adopted via a contract with the firefighter organization. The agent can distinguish between different rules based on their origin. It knows that if it gives full priority to the organizational rules, it follows the organizational norms. It still has the possibility to prefer its own event-processing rules; however, this may lead to violation of organizational norms.

In Chapter 2 situations are described where organizations benefit from violation of norms. An agent's local observations can conflict with organizational rules. For example, if a firefighter feels that he is in danger he could ignore a request from the coordinator and give priority to his own goals in order to stay safe. He deliberately gives priority to his own goals, and therewith risks violating the organizational norm.

The ability to distinguish between the reasoning rules can be used for prioritization of the rules. This can be done by applying machine-learning techniques. A learning agent can learn the situations in which specific rules should have priority, such as the danger-example.

Another option is to prioritize based on heuristics. Prioritization is studied in argumentation logics [Brewka, 1994]. Argumentation as been applied reason about interaction between agents [Kakas et al., 2005]. Our rule-based approach of event processing fits very well with this type of metareasoning. We describe an example in the next paragraph.

Violations

The agent can use metareasoning via prioritization to handle norm violations. Imagine that the agent has the limitation that he can only extinguish one fire at the same time. It receives a request to extinguish fire $F1$ and it adopts the

task. While it is busy extinguishing the fire, it receives a request to extinguish fire $F2$. Immediately accepting the second request would lead to a violation of the first, since $F1$ has not been extinguished yet. The agent knows that the violation of the first request stems from rule $r1$ that gives meaning to the hierarchical relation. The agent might as well have some personal preference to finish tasks before taking up new ones:

```
message(SENDER, request, TASK) :- busy() | IgnoreEvent()
```

However, this rule would lead to a violation of the request to extinguish $F2$. Both rules $r1$ and $r4$ are applicable.

Several prioritization rules can be thought of to handle those violations. One can judge the requests and choose the violation that is least bad. For example, *fire F1 is more serious that fire F2, and therefore I violate the second request, so I apply rule r4.* Or one can judge the rules that give rise to the violations. For example, *General interaction rules are more important than personal preferences, therefore I apply rule r1 and I will violate the request to extinguish F1.*

Attitudes

Situation-based prioritization is a type of metareasoning. Given that an agent can reason about norms, it can use metareasoning to take different attitudes with respect to the organization. In the example described above, the agent has to choose between two violations. One of the solutions proposed, gives priority to a specific type of event-processing rule. This represents the attitude of an agent with respect to the organization. In the example, the agent gives priority to the organization.

Sichman and Conte [Sichman and Conte, 1998] and Dastani et al. [Dastani et al., 2004b] describe several possible attitudes. For example, an agent can adopt the organizational goals and drop its private goals, or it can still prefer its own goals above the organizational ones. The agents' attitudes have an effect on the performance of the organization. McCallum et al. [McCallum et al., 2008] describes organizational change in terms of influences between the roles. The model we presented in this chapter allows the actual implementation of agent types with different attitudes.

5.7 Conclusions

Agents in an organization need to coordinate their actions in order to reach the organizational goals. Organizational models specify the desired behavior in terms of roles, relations, norms and interactions. However, the actors in an organization are autonomous entities that control their internal state and their behavior. In this research we have shown how organizational rules can be adopted by autonomous agents. We have developed a way to translate norms into reasoning rules for event

processing. The reasoning rule contains a trigger, situational constraints and an effect on the agent's mental state that are derived from the norm specification.

We have proposed a modular reasoning model to make organizational rules explicit. Since the agents are autonomous entities they will have their own reasoning rules next to the organizational rules. The modular approach makes that the agents can distinguish between different event-processing rules and that they are aware of the organizational rules. This allows for metareasoning about event processing, and gives the agent control over its internal state. It guarantees the autonomy of the agent and at the same time makes group coordination possible.

An important aspect of our method is the translation from the norms to reasoning rules of the agent. We have used a simple example of a firefighter organization to illustrate our ideas. More complex organizations might introduce complex behavioral rules and we have to evaluate whether we can express them in our language for event-processing rules. Furthermore, we argued that the possible effects on an agent's mental state are dependent on its decision-making process. This has implications for the translation as well.

We have presented some advantages of using modularity in the reasoning model. In our approach, the agent can do metareasoning reason about the event-processing rules, and thus about norms. We have used prioritization of event-processing rules as example of metareasoning. In Chapter 6 we investigate different methods of metareasoning, such as argumentation-based techniques.

Chapter 6

Adjustable Autonomy: Reasoning about Influence

In order to achieve optimal results, an agent's way of decision making might need to change according to the circumstances. One of the aspects an agent can adapt is the way it processes external events, such as observations and messages. We argue that influence control is a form of metareasoning. In this chapter, we describe heuristics for event processing that specify the agent's openness towards others and to the environment. We propose a model for metareasoning that allows the agent to select and prioritize those heuristics. The model includes a component to measure performance based on goal achievement, a selection mechanism to select desired heuristics and a control mechanism that does the activation and prioritization of the heuristics. We describe how to specify these components to obtain the required adaptivity. Finally, we discuss the agent's performance in a scenario involving crisis situations.

6.1 Introduction

Actors in dynamic environments continuously perceive many stimuli via observations and messages. This research provides a reasoning model with which actors have the capability to deal with external influences in an adaptive way.

Consider the dynamics and complexity of crisis situations, for example a fire brigade sent out to extinguish fires. The firefighters are able to communicate with each other via speech acts over a radio channel. Not all conversations over the radio are relevant for all firefighters. For instance, when firefighters report status updates over the radio channel all firefighters can hear the information, but it is only relevant to the ones in the same location. When a firefighter is busy extinguishing he does not have time to listen to all conversations. If he tries to follow all communication, this will take up his mental capacity and therewith

influences his ability to focus on his task. The solution is that he evaluates very quickly whether a conversation is relevant or not. The firefighter makes this relevance decision based on its tasks and its status. In this case, he will focus on the observations concerning the fire he is extinguishing.

We can generalize this example. Within an organization, information distribution is important. Requirements of information distribution are guaranteed via communication protocols. However, it is impossible to know at the organizational level exactly which information is relevant for the individual actors. At the same time, leaving out crucial information may have unwanted consequences. Therefore, the protocols take care of general information distribution and the actors need to determine locally whether information is relevant with respect to their tasks. With this local process of relevance determination, the actor controls how its decision-making process is influenced by new information.

A similar process occurs with respect to actions. The organization coordinates the behavior of the actors via norms or policies. However, the organization can not control the actions. At the organizational level it is not possible to specify the desired actions for all situations. Actors can encounter situations where different norms, possibly from different organizations, conflict which each other. Actors have to make their own decisions, where they can use the organizational norms as guidelines for their decisions.

The example shows that autonomous actors actively control to what extend their decisions are influenced by the environment and by others. This implies that an agent adapts its reasoning by changing the way external events are processed. Adaptation of the reasoning process requires a feedback loop that monitors and changes the reasoning process. Therefore, it is a form of metareasoning.

Metareasoning is reasoning about reasoning. A general model for metareasoning consists of a performance measure of current reasoning, a process to select a new reasoning strategy and the adaption from one reasoning strategy to the other [Cox and Raja, 2008]. The adaptation can apply to different aspects of the reasoning process: knowledge representation [Cox, 2005], the deliberation cycle to decide on actions [Dastani et al., 2004a], or resource allocation [Hansen and Zilberstein, 2001]. We argue that event processing is part of reasoning as well, which can be subject of metareasoning.

In this chapter we focus on controling external influences as a form of metareasoning. There are several heuristics the agent can use to deal with external influences. The heuristics are based on different types of knowledge: introspective knowledge such as information relevance [Castelfranchi, 1995], social knowledge such as trust [Barber and Kim, 2001], or organizational knowledge [Vecht et al., 2009]. We combine the heuristics for influence processing and allow the agent to reason about the use of the heuristics. We apply the general model for metareasoning to influence processing.

In our model, the activation of the heuristics specifies the attitude of the agent towards the environment and towards other agents. The agent can switch between different attitudes by activating, ignoring and prioritizing the heuristics.

The mechanism to decide upon the best attitude is based on measurable features of the agent's goals.

The chapter is structured as follows; Section 6.2 discusses other research on metareasoning. It describes a general model for metareasoning and discusses the aspects of object-level reasoning that can be subject to the metareasoning process. Section 6.3 discusses a reasoning component to process external events, and it discusses how we apply the general metareasoning model to this component. In Section 6.4 we explain the process of metareasoning in a detailed way. Section 6.5 gives an example of metareasoning using the model. In a scenario in Section 6.6 we show how agents react adaptively to changing circumstances and we show the benefits for organizational performance. In Section 6.7 we conclude our work.

6.2 Background

The scenario from the introduction shows that the process of dealing with external influences can be adjusted to control the performance of the decision making. Therefore, it is a form of metareasoning. In this section we discuss a general model for metareasoning. We describe different approaches of metareasoning, and we discuss the relation with other work on adjustable autonomy.

6.2.1 A General Model for Metareasoning

Traditionally reasoning is seen as a decision-making cycle within an action-perception loop. An agent perceives stimuli from the environment and behaves rationally to achieve its goals by selecting the appropriate actions from its set of capabilities. Metareasoning is the reasoning process about this object-level reasoning.

The idea behind metareasoning is that there is not one perfect way of decision making in complex dynamic environments. The goal of metareasoning is to improve the quality of the agent's decisions by spending some effort on deciding what and how much reasoning to do, instead of looking at what actions to take. The agent performs its actions at the ground level. At the object level the agent reasons about which actions to take. Metareasoning reasoning is reasoning about the object level. The metareasoning process can be seen as a parallel process to object-level reasoning.

When creating a model of metareasoning, the process should be defined in terms of components. What internal actions does an agent perform when it is reasoning about its reasoning? Given that an agent has or can construct several reasoning strategies, earlier work on metareasoning distinguished the following components [Cox and Raja, 2008]:

- A **monitor** process that monitors the current reasoning strategy
- A **performance measure**-component to evaluate the performance of the current reasoning strategy

- A **selection mechanism** to choose between different available strategies
- A **control mechanism** to handle and switch between reasoning strategies

An agent needs different strategies of reasoning. In order to determine which reasoning strategy to take, an agent needs a performance measure to compare different strategies. The agent should notice when a certain strategy fails and it should select a new strategy. The meta-level control mechanism describes how to switch from one strategy to another. Figure 6.1 shows the components of the metareasoning process.

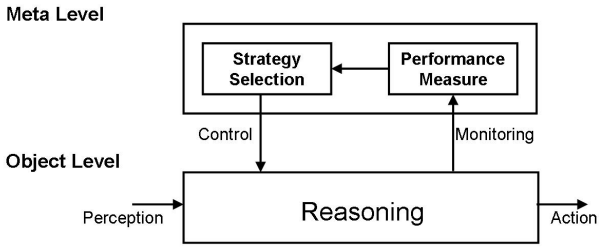


Figure 6.1: *The general model for metareasoning*

6.2.2 Object-Level Reasoning

The metareasoning process requires that the agent is able to distinguish several aspects of the object-level reasoning. Which aspects of object-level reasoning can be changed? What is the influence of adaptation of a certain component on the reasoning process?

We are assuming autonomous agents, as we have argued before. In Chapter 2 we have defined being autonomous as having control over external influences on the decision making process. We have taken an abstract look at object-level reasoning, we distinguish these two main aspects of reasoning:

- controling external influences
- Decision making.

An agent should control to what extend external events take an effect on its mental state. Therefore, the agent needs a component to process incoming events. For example, an agent determines whether or not the sender of a message can be trusted before adopting the content of the message. The second part concerns the actual decision-making process of the agent. Based on its internal state, the agent decides which actions to take in order to fulfill its goals. Both processes rely on the agent's mental state. Therefore, the mental state is a third important

component. In Chapter 3 we have described the object-level reasoning process in more detail.

Figure 6.2 shows the components of the object-level reasoning process. All three aspects, the process of influence control, the mental state and decision-making process, can be subject of metareasoning.

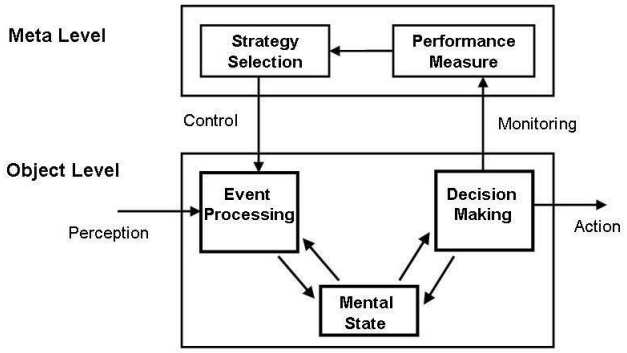


Figure 6.2: *The reasoning components at object-level and meta-level*

By adjusting the event-processing module, the agent chooses *how* and *by whom* its internal state is being influenced. Adjustment the event-processing module is when an agent changes its attitude towards the environment and towards other agents. For example, the agent decides to focus on specific information it needs for its task, and as a consequence ignores (some) other observations. The way external events are processed has influence on the performance of the agent. We use adaptation of event processing to optimize the agent's performance. Therefore, the model needs to have a feedback loop from the decision-making component to the event-processing component. Monitoring the performance of the current decision-making process may lead to adaptation of the event processing, as Figure 6.2 shows.

6.2.3 Perspectives on Metareasoning

An overview of research on metareasoning is given by Cox [Cox, 2005]. As stated, several aspects of object-level reasoning can be subject of metareasoning. In figure 6.2 we have distinguished three components at the object-level: the mental state, event processing, and decision making.

The mental state can be subject to metareasoning. For example, the way how knowledge is represented can be adjusted. This aspect has been studied in the logic community [Cox, 2005]. The agent can reason about its knowledge. The metareasoning process then determines the way object-level knowledge is represented.

The decision-making process can be adjusted as well. An intuitive problem is discussed in [Hansen and Zilberstein, 2001]. The paper discusses anytime algorithms, which generally have a trade-off between solution quality and computation time. The metareasoning consists of deciding whether to stop the deliberation and act, or continue the deliberation.

Metareasoning is discussed as well in the BDI-reasoning approach, where agents reason about their beliefs, goals and intentions to decide on the next action. BDI-deliberation consists of a series of mental actions. Dastani, et al., [Dastani et al., 2004a] analyze deliberation for BDI-agents. The researchers propose a meta-language to describe the deliberation cycle of an agent. The language allows the construction of different deliberation cycles. For instance, changing the order of the mental actions can adjust the frequency of reconsidering previous choices. In their model, metareasoning would result in adjustment of the deliberation.

Related to metareasoning are studies on emotional agents. Lim et al. [Lim et al., 2009] propose an agent architecture, where the generation of intentions is depends on local measures such as competence, certainty and urgency. The local measures together represent an attitude based on which intentions are generated. This way, the researchers connect BDI-reasoning to dynamic intention generation. We aim for a similar control process, but instead of reasoning about intentions, we apply it to influence control.

In this chapter, we focus on metareasoning about the event-processing module. This module determines to what extend the agent is being influenced by external factors. The openness towards other agents and towards the environment can be adjusted according to the performance. Metareasoning results in a level of openness.

If we think about real-world species, animals or humans, they have mastered all forms of metareasoning in some way or another, as well as the interaction between them. The related work on metareasoning we described above discusses knowledge representation as well as the decision-making part. In this work, we focus on metareasoning for influence control.

6.2.4 Adjustable Autonomy

By adjusting the process of event-handling, the agent controls its openness towards its environment and towards others. Openness is strongly related to autonomy. The autonomy level of an agent with respect to a decision can be related to the degree in which other agents interfere with the decision-making process [Martin and Barber, 2006]. The more other agents interfere with the decision-making process, the lower is the autonomy level of the agent with respect to the decision, and vice versa.

Studies on adjustable autonomy as well adjust the decision-making process in order to increase performance. One model for adjustable autonomy is given by Bradshaw et al [Bradshaw et al., 2005]. The researchers study the autonomy of a system with respect to a user. By changing the abilities, permissions and

obligations of the system the user controls the system's freedom of choices. Just like in metareasoning models, the decision-making process is adjusted. However, the difference is that within metareasoning models the agent has the control over its decision-making process, whereas Bradshaw, et al. give the control to an external actor.

Another approach to adjustable autonomy can be found in [Tambe et al., 2002]. In their work, adjustable autonomy refers to the process where the control over a certain decision is transferred from one agent to the other. Adjustable autonomy is used as a coordination paradigm in the context of multi-agent coordination.

In contrast to the other approaches, we focus on optimization of single-agent decision making. Our work concerns on self-adjustable autonomy. The agent adjusts its openness towards the environment and towards others in order to increase its performance. By allowing or disallowing influences the agent controls the autonomy level of its decision-making process. Following the definition in Chapter 2, we consider the process of dynamically controling external influences as adjustable autonomy.

6.3 Influence Control

The object-level reasoning model consists of two main processes: influence control and decision making. In this section we show the extension of object-level reasoning with a metareasoning component. We describe general heuristics that we use to process external events. The next step is to reason about the heuristics.

6.3.1 General Heuristics for Influence Processing

Here, we consider the module for influence control as used in object-level reasoning. One can think of several heuristics that can be used to process external events that together determine how and by whom the agent's mental state is being influenced.

An agent can control *how* it is being influenced for instance by processing new information based on relevance with respect to current goals/tasks. The use of information relevance to control the autonomy of an agent has been discussed by Castelfranchi [Castelfranchi, 1995] and in Chapter 4 of this thesis. It allows the agent to focus on specific information for a certain task. The agent can protect itself from information overload.

Another heuristic is to process information based on knowledge about organizational or social context. The agent then controls *by whom* it is being influenced. Does a request originate from a superior or from an unfamiliar source? Agents achieve coordination by allowing influence on the internal state based on social and organizational knowledge. Chapter 5 of this thesis describes how organizational rules can be translated into reasoning rules for event processing. If an agent follows the event-processing rules of an organization, it knows that it fol-

lows the norms of the organization. A similar approach is taken by Bradshaw et al., [Bradshaw et al., 2005]. In their framework, organizational policies determine the autonomy of the agent by restricting optional and permitted actions.

Other reasons to control by whom an agent is being influenced can be socially inspired, for example *trust*. Can the sender of a message be trusted? Trust can be defined as the acceptance of a new statement without further evidence. Several models for trust have proposed. A model for trust is presented by Bosse et al [Bosse et al., 2007]. The researchers consider trust states as a type of mental states and they analyze the dynamic properties of trust. Barber et al., present a model for internal belief revision based on the trust level of the other agents [Barber and Kim, 2001]. The mechanism to determine the reputation of agents counts as the social context in the model. The use of trust in heuristics to process messages can be seen as social knowledge.

From the above we construct three different heuristics for influence processing:

1. Only accept information that is relevant with respect to current goals/tasks
2. Process messages/observations based on organizational norms
3. Accept messages only when the sender is trusted

The heuristics have a very general character and do not contain domain-specific knowledge. They can be implemented in the component for event processing at the object-level. The knowledge and meta-knowledge that is used in the heuristics should be derived directly from the mental state of the agent. Therefore, the agent needs to be able to derive relevance of information. The social and organizational knowledge needs to be present in the knowledge base.

For the remainder of this work we continue with these three heuristics. However, it might be possible to construct other domain independent heuristics for event processing.

6.3.2 Attitudes

The use of heuristics for event processing results in a certain degree of openness towards the environment and to other agents, as they specify the effect of external events on the mental state, and thus on the decision-making process. We can classify the heuristics from Section 6.3.1 in terms of their attitude towards the agent's environment, listed in Table 6.1. An agent can follow or ignore a heuristic. The table shows the result in terms of attitude.

An agent that only accepts relevant information for a certain task is focused on that task. An agent that does not care about the relevance heuristic is open for all information.

An agent that follows the organizational norms is obedient, whereas if it is ignorant for the norms it acts opportunistic. The opportunistic agent selects its action only based on its profit, and does not care whether it obeys or follows a norm.

Table 6.1: *Classification of heuristics in terms of the attitude towards the agent's environment*

Heuristic	**Following the heuristic**	**Ignoring the heuristic**
Relevance	Focused	Open
Norms	Obedient	Opportunistic
Trust	Uncertain	Confident

Trust is an issue for the agent, if it lacks confidence in a task and needs to rely on others opinion. If an agent has confidence in its own ability to weigh the truth of a statement, trust is no issue and the heuristic can be ignored.

If we consider these three heuristics we can create eight strategies by either activating of deactivating each of the heuristics. The set of active heuristics determines the overall attitude of the agent towards the environment and towards other agents. The heuristics determine how and by whom the agent can be influenced. Therefore, the heuristics define the level of autonomy of the agent's decision-making process.

6.3.3 Influence Control via Metareasoning

The set of heuristics that is active determines the overall attitude of the agent towards the environment and therewith the autonomy level of its decision-making process. However, if an agent always follows the same heuristics, it processes events always in the same manner. Having control over external influences implies that the agent dynamically selects heuristics for event processing based on its situation.

At the metalevel of the reasoning process the following type of questions need to be answered:

- Is relevance an issue when processing observations?
- Are the organizational norms valuable?
- Is trust an issue when processing messages?

Dealing with these questions is the task of the metareasoning component. At the meta-level, the agent should decide which of the available heuristics it uses. In the next section we discuss the metareasoning model.

6.4 Metareasoning Model

We propose to use the heuristics as building blocks for attitudes of the agent. The heuristics can be used in the metareasoning model. In this section we describe

how monitoring the object-level results in performance measures, which serve as the input of the selection mechanism and we describe the control mechanism to switch between heuristics.

6.4.1 Performance Measure

The performance measure of the agent serves as input for the selection mechanism. As we are dealing with pro-active agents that are pursuing a certain goal, we can measure the performance based on goal achievement.

Choices on the selection mechanism require different types of performance measures. We present general triggers that can be used to select event-processing heuristics. These triggers are domain-independent. We assume that the agent only needs to change its reasoning strategy if it fails to achieve its goals. As we monitor the agent's goals, we define triggers that consist of goal properties and relate them to the openness feature of the heuristics.

Introspection

The relevance heuristic allows focusing on a specific task. When does an agent want to focus on a task, and when is it not necessary to focus? An intuitive trigger to turn on or off the relevance heuristic is *task urgency*. When the urgency gets high, the agent might decide to focus on that goal, and allow only relevant information for that task into its mental state. When a goal has no urgency, the agent can decide to take up some other tasks in the meantime.

Organizational Knowledge

With respect to organizational knowledge, the agent can distinguish the goals based on their origin. The goal or task can be part of an organization, or it can be a goal of the agent itself. An agent can be part of several organizations, and every organization has norms with corresponding reasoning rules for influence control. If the agent actively pursues a goal of a certain organization, the agent should follow the norms of that organization. The loyalty of an agent with respect to the organization gives an indication of the dedication with respect to the goal. Therefore, the goal origin can be related to a dedication measure.

Social Knowledge

The trust heuristic reflects the social knowledge of the agent. The agent acts more autonomously if it only believes others it trusts, as it limits the influence of other agents. When the agent believes everyone, everyone can influence the decision-making process of the agent. Implementations of current trust models make that agents use trust estimations for their reasoning at any time. The trust model presented in [Bosse et al., 2007] studies the dynamics of trust that result from past experiences. However, the dynamics are part of the trust model itself.

Table 6.2: *Heuristics for influence processing, the related goal feature and the adjustment type*

Heuristic	Goal Feature	Adjustment
Relevance	Task Urgency	(De-)activation
Origin	Dedication	Prioritization
Social Knowledge	Confidence	(De-)activation

The agents in their model do not address the question whether trust is an issue at all, which is the question we want it to address. At the meta-level we should discuss why does an agent decide to only deal with agents it trusts? And why is it not selective in believing others?

The trigger we use here is *confidence*. If an agent is very confident, it relies on its own interpretation of information and whether the trust level of others is not an issue. This is the case, for example, when an agent considers itself an expert. When the agent has low confidence and is uncertain about its ability to value new information, it relies on the trust level of the other agents. This can be the case when the agent is communicating with an expert.

6.4.2 Selection Mechanism

Given a performance measure, the selection mechanism needs to choose to become more open of more closed towards the environment and other agents. The selection mechanism is provided with a prioritized set of goals, with the distinguishable features *task urgency*, *dedication* and *confidence*. The features of the goals are linked to reasoning strategies. Table 6.2 shows the general relations.

The output of the selection mechanism is the activation deactivation of the heuristics. The dynamics of the relevance and trust heuristic are shown in Figure 6.3.

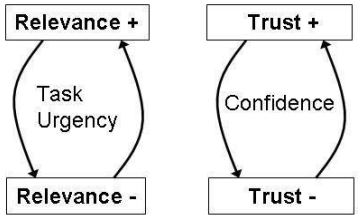


Figure 6.3: *Heuristics and their triggers*

The selection mechanism selects as well a priority order of the heuristics based on the origin of the goals. We translate the origin of a goal to a dedication mea-

sure. If the agent actively pursues a goal of a certain organization, the agent should follow the norms of that organization. However, if the agent has a higher dedication towards another goal from a second organization, the agent gives priority to the norms of the second organization. The result of the origin measure will be a prioritization of norms and thus a prioritization of heuristics for event processing.

6.4.3 Control Mechanism

The control mechanism translates the result of the metareasoning process to the object-level reasoning. The component for event processing should be adjustable, such that the desired attitude is taken. This means that the activation and deactivation of heuristics is carried out and the prioritization of the heuristics.

6.5 Agent Specification

So far, we have presented the components of a meta-reasoning model with which the agent gets control over external influences. The model is domain-independent; the heuristics for event processing are general heuristics. The performance measure is specified in general goal features and the selection mechanism uses those features as triggers for the heuristics. We have defined metareasoning components independent from the decision-making model.

In order to make the model for metareasoning work, we need to specify how the agent maintains the list of its goals, including the goal features and how the triggers influence the heuristics. The designer of the agent has several design choices here that relate to the implementation of the object-level. In this section, we show one way to implement all object-level components and meta-level components. We operationalize the performance measure and the selection mechanism further, and we show how design choices have consequences for the dynamic properties of the agent.

6.5.1 The Object Level

Here, we specify the choices on object-level reasoning and we present optional solutions.

Event Processing

The event-processing module translates external events to effects on the internal state of the agent. Here, we briefly describe a rule-based solution for this component.

The component for event processing implements the heuristics described in section 6.3. The heuristics can be translated to reasoning rules of the form:

```
<event> <- <constraints> | <effect>
```

The event is a message or observation that reaches the agent, the effect is the effect on the mental state that results from the event. The constraints specify situational constraints to which the effect is restricted. The heuristics for relevance and trust can be defined as follows:

```
observation(X) <- relevant(X,G) | adopt(X)
message(Sender, inform, X) <- trusted(Sender) | adopt(X)
```

In the case of an observation, the effect on the mental state can be the adoption of the observed information or the observation can be ignored. A message can result in the adoption of ignorance of the content.

The example rule above contains the predicate $relevant(X, G)$, that describes the relevance of a piece of information X with respect to goal G. The agent needs to be able to derive this predicate. Several models of relevance determination exist for different contexts. In Chapter 4 we introduced an algorithm for relevance determination in BDI-agents.

The second reasoning rule contains the predicate $trusted(S)$, which specifies whether the sender of the message, agent S, can be trusted or not. Therefore, it is required that the agent has a model to determine whether others can be trusted. Several models of trust exist, based on memory or based on social interaction. Example of trust models are presented by Barber et al. [Barber and Kim, 2001] and Bosse et al. [Bosse et al., 2007].

Organizational norms can be translated into reasoning rules that have deontic results, such as obligations, permissions or prohibitions as the resulting effect of an event. We described this translation process in Chapter 5.

The reasoning rules for event processing are maintained in an ordered list. The rules can be activated or de-activated individually. The reasoning rules can result in contradicting effects on the mental state. For example, based on the relevance rule a certain observation should be ignored, whereas based on an organizational norm, the observation should result in the task to communicate the information to the superior of the agent. In such a contradicting case, the agent follows the rule with the highest priority.

Mental State and Decision Making

For the mental state and the decision-making model we take the BDI approach. The mental state of BDI agents contains beliefs, goals and plans. BDI reasoning is logic based. The knowledge in the belief base can be represented in semantic relation via deduction rules.

The agent reasons about the concepts in the decision-making process to decide upon actions. Several BDI implementations exist that can be used for the action deliberation, for example, AgentSpeak(L) [Rao, 1996] or 3APL [Dastani et al., 2004b].

6.5.2 The Meta Level

Here we describe implementation solutions for the reasoning components at the meta level.

Performance Measure

The component to measure the performance has as output a list of goals of the agent. All goals have features that describe their urgency, their dedication measure and the confidence as shown in Table 6.3. The agent needs to be able to calculate those features.

We propose to maintain the goal features in the belief base of the agent. We can define a predicate *haveGoal*(), that is derived using deduction rules. The requirements to derive *haveGoal* are that the agent has added the goal in the past. When the agent adds a goal, it knows the origin; whether its an own goal or and organizational goal. The dedication, urgency and the confidence can change over time. The agent needs functions to calculate those values. As a result, we define the *haveGoal* predicate as follows in a Prolog-like manner:

```
haveGoal(Goal, Origin, Urgency, Dedication, Conf) :-
    addedGoal(Goal, Origin),
    calcUrgency(Goal, Urgency),
    calcDedication(Goal, Dedication),
    calcConfidence(Goal, Conf).
```

A possibility to calculate the urgency is to take the time the agent has been pursuing the goal as measure. However, more complex measures are possible. It is not unlikely that domain-specific knowledge plays a role in determining the urgency of a goal. As an agent has been designed for a certain purpose, the designer might increase the urgency of the goals within the purpose domain. For example, a firefighter can make a goal to survive more urgent when explosion danger increases.

Confidence can be calculated in a similar way. Confidence is a measure of estimated success. The confidence of an agent with respect to a certain task can starts at a certain value, which is possibly predefined. This value can decrease with the number of failed attempts and increase based on successful ones. Domain-specific knowledge can influence the confidence as well, for example, by looking at availability of resources.

Table 6.3 gives an overview of the values that can be returned by the performance measure. Here we assume that the task urgency and the confidence is calculated as a value between 0 and 1. The table shows an example of a firefighter with two goals; one to stay safe, and one to inform other agents about its status.

Selection Mechanism

The selection mechanism relates the output of the performance measure to the activation and deactivation of heuristics and the priority order.

Table 6.3: *The goal features that are used as performance measure and their values.*

Goal	**Origin**	**Urgency** $[0,1]$	**Origin** $[0,1]$	**Confidence** $[0,1]$
$Inform$	$FireBrigade$	0.8	0.8	0.8
$StaySafe$	$Self$	0.8	0.6	0.8

We defined task urgency as a trigger to activate the relevance heuristic. We need to define a threshold n. If a goal G has urgency larger then n, the relevance rule is activated:

```
observation(X) <- goal(G) AND relevant(X,G) | adopt(X)
```

The designer should set the threshold required to activate the rule. A low n implies a character that focuses on its goals and does not get distracted by other information. A high n implies an agent who is open to environmental influences.

We need to define a threshold for the confidence that activates or deactivates the trust heuristic. The agent has a variable c, which is a confidence measure. If a goal G has an associated confidence lower then c, the trust rule is activated:

```
message(S, inform, X) <- trusted(S) | adopt(X)
```

The designer should set the threshold. A low c implies a character that easily falls back on the trusted actors in its environment. A high c implies that the agent does not find trust an important feature and thinks it can rely on its own opinion.

A set of reasoning rules contains the norms of an organization O. If organization O is the origin of one of the goals, the corresponding norms are to be followed and thus the reasoning rules are active. The priority of the event-processing rules reflects the dedication of the agent with respect to the organization. The rules with the same origin as the goal with highest dedication, receive the highest priority. The rules corresponding to the goals with lowest dedication, receive the lowest priority.

The output of the selection mechanism is an ordering of the list of reasoning rules for influence processing and the activation for each of the rules. The control mechanism applies this at the object-level.

6.5.3 Overview

We have described a reasoning model for agents at object-level and metalevel. The object-level consists of two main processes: external event processing and decision making. The event-processing module contains reasoning rules to process observations and messages. The active rules specify the attitude of the agent with

respect to the environment and other agents. The decision-making process is the actual action-selection process.

At the metalevel the agent reasons about the object-level. In our case it reasons about the event processing and it adjusts its attitude with respect to the environment and other agents. The meta-level consists of four processes: *monitor object-level*, *determine performance*, *select desired attitude*, *adjust current attitude*.

Meta-level Reasoning

Determination of the desired attitude of the agent requires that we monitor and value the current performance of the agent. We determine the performance of the agent by analyzing the goal achievement of the agent. The goals of the agent can be distinguished based on several features: *urgency*, *dedication*, and *confidence*.

The monitoring process requires some measuring skills of the agent. The agent needs methods for estimation and calculation. We propose the following methods to calculate urgency, dedication and confidence:

- Urgency (value between 0 and 1) based on deadlines and time of trying
- Dedication (value between 0 and 1) based on origin (self, organization)
- Confidence (value between 0 and 1) based on experience

Based on these features we monitor the agent's current goals. The selection mechanism to determine the desired attitude uses the prioritized goal list to come to a desired attitude. We have three event-processing heuristics that can be activated or de-activated: *information relevance*, *organizational knowledge* and *trust*.

If the information-relevance heuristic is used, the agent only adopts relevant information with respect to its goals. The organizational-knowledge heuristic specifies the way the agent has to react according to the applicable norms and organizational rules. The trust heuristic determines whether the agent relies on beliefs of other agents or not.

We define meta-heuristics that link the performance measure of the agent to the event-processing heuristics:

- Based on the urgency of goals, the agent decides on the relevance heuristic. If the urgency of a goal exceeds a threshold, the relevance heuristic of this goal becomes active.
- Based on the dedication with respect to the goals, the importance of norms of corresponding organization is made, which determines the priority order of event-processing rules.
- Based on its self-confidence measure, the agent determines whether trust in an issue or not. If the confidence goes below a threshold, the agent needs to rely on others and trust becomes important.

The meta-heuristics are used in the selection mechanism to determine the new attitude of the agent. The meta-heuristics presented above require that thresholds are defined to trigger the event-processing heuristics. For each of the heuristics we need to define base values:

- Urgency threshold
- Dedication threshold
- Confidence threshold

The final process of the metareasoning is the adjustment of the object-level reasoning. In the model presented here, the agent reasons about event processing. The output of the meta-reasoning process is a prioritized list of event-processing rules. The control mechanism adjusts the object-level by changing the event-processing module according to the chosen event-processing rules. The result is a new attitude of the agent with respect to the environment and to other agents.

In Figure 6.4 we give an overview of the general and specific components of the reasoning model for both the object level and metalevel.

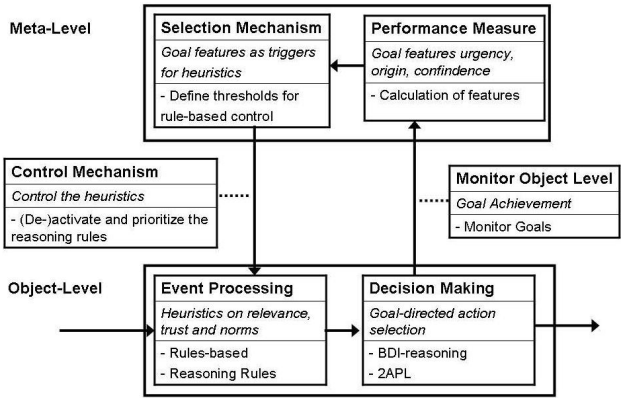


Figure 6.4: *An overview of the reasoning model with separate* ***components****, the* general solution*, and the implementation choices*

6.6 Experiment

Here we demonstrate the use and the effect of the meta-heuristics on the adaptivity of the agents. We use the experimental setting of the firefighter organization as described in the several chapters before.

There are two roles; coordinator and firefighter. The coordinator has a global world view. He monitors the fires in the world, and makes a planning of which

ID	Origin	Event	Rule
ep1	Default	Observation	`Observation(X) <- TRUE | UpdateBeliefs(X)`
ep2	Default	Inform Message	`Message(S,inform,X) <- TRUE | UpdateBeliefs(X)`
ep3	Default	Request Message	`Message(S,request,X) <- TRUE | AddGoal(X)`

Table 6.4: *Default event-processing rules for self-reliant, trusting, cooperative agents*

firefighter should extinguish which fire. He can send requests to the firefighters. There is a hierarchical relation between the roles; the coordinator is superior of the firefighters. This implies that according to the organizational rules, the firefighters have to accept the requests of the coordinator.

The firefighters have as goal to extinguish fires. They can observe the world locally, move around in the world and perform extinguish actions. They receive messages from the coordinator with the request to extinguish a specified fire. According to the organizational rules they are obliged to accept the request.

The firefighters have another goal, which is to stay healthy. In order to stay healthy, they need to take a rest from time to time. They experience their fitness in the form of observations. Their fitness level decreases slowly when they are busy performing tasks, and it increases again if they take a break.

The possible observations of a firefighter are: fires in the direct environment, tiredness level and status. The firefighter can observe three types of status: *ready*, *busy*, or *paused*. Furthermore, the firefighter can receive messages from other agents, including the coordinator.

6.6.1 Default Behavior

In this experiment we focus on the firefighters. We analyze how the metareasoning process influences the behavior, both of the individual and the organization. First we give the implementation of the object-level reasoning and we show the behavior of the agent without metareasoning. We start with the module for event processing, and next the decision-making module.

event processing

The event-processing rules specify how the agent observes the environment. Based on the attitudes discussed in Chapter 3, we assume self-reliant, trusting, cooperative agents. Table 6.4 sums up the default event-processing rules that follow from this basic attitude.

Furthermore, there are five event-processing rules that motivate the agents goals, based on own preferences and the organizational rules. These rules generate the goals of the agent. The own preferences describe motivations to extinguish

ID	Origin	Event	Rule
ep4	Self	Observation	`Observation( Fire ) <- TRUE | AddGoal( ExtinghuishFire(Fire) )`
ep5	Self	Observation	`Observation( tiredness ) <- TRUE | AddGoal( TakeBreak() )`

Table 6.5: *Agent's event-processing rules for goal generation*

ID	Origin	Event	Rule
ep6	Firefighter	Observation	`Observation( StatusChange ) <- TRUE | AddGoal( InformAboutStatus() )`
ep7	Firefighter	Request Message	`Message( Coordinator, request, X) <- TRUE | AddGoal( X )`
ep8	Firefighter	Request Message	`Message( Coordinator, request, X) <- TRUE | AddGoal( AnswerRequest() )`

Table 6.6: *Event-processing rules for goal generation of Firefighter role*

a fire when the agent observes one and to take a break when he is getting tired. The two rules describing the own preferences are listed in Table 6.5.

The coordination mechanism of the firefighter organization includes rules for information distribution; a firefighter is obliged to inform the coordinator about its status. Furthermore, a firefighter is obliged follow a request from the coordinator and to reply on a request message from the coordinator. The organizational rules are translated into event-processing rules that generate goals from incoming events. The three rules following from the organizational norms are listed in Table 6.6.

These eight event-processing rules are the set that determine the agent's standard behavior without metareasoning. We give the standard prioritized list of active event-processing rules in Table 6.7, where a lower number indicates a higher priority. The priority is based on the origin of the goals. The organizational rules have the highest priority, followed by the rules that contain the own preferences, and finally the default rules are applied.

When we include meta-reasoning, the agent can construct event-processing rules based on information relevance and trust. They are listed in Table 6.8. The relevance heuristic consists of three rules, $ep9$ to $ep11$, describing that only relevant observations are passed on and all other events are ignored. The trust heuristic is captured by rule 12. Based on dedication to organizations, the agent can prioritize the list of event-processing rules. The prioritized, active rules will influence the behavior of the firefighter agent.

Priority	ID	Origin
1	ep6	Firefighter
2	ep7	Firefighter
3	ep8	Firefighter
4	ep5	Self
5	ep4	Self
6	ep1	Default
7	ep2	Default
8	ep3	Default

Table 6.7: *Default set of active, prioritized event-processing rules*

ID	Origin	Event	Rule
ep9	Self	Observation	`Observation(X) :- relevant(X, G) | AddBelief(X)`
ep10	Self	Observation	`Observation(X) :- NOT relevant(X, G) | IgnoreEvent()`
ep11	Self	Message	`Message(S, P, X) :- NOT relevant(X, G) | IgnoreEvent()`
ep12	Self	Inform Message	`Message(S,inform,X) :- trusted(S) | UpdateBeliefs(X)`

Table 6.8: *Event-processing rules based on heuristics*

ID	Goal	Rule
dm1	TakeBreak	`TakeBreak() <- tiredness | Pause`
dm2	ExtinguishFire	`ExtinguishFire(X) <- Distance(X) > 10 | GoTo(X)`
dm3	ExtinguishFire	`ExtinguishFire(X) :- Distance(X) < 10 | Extinguish(X)`
dm4	AnswerRequest	`AnswerRequest( Message(Sender, request, X) ) <- goal(X) | Send(Sender, accept, X)`
dm5	AnswerRequest	`AnswerRequest( Message(Sender, request, X) ) <- NOT goal(X) | Send(Sender, reject, X)`
dm6	InformAbout Status	`InformAboutStatus <- status(X) | Send(coordinator, inform, status(X) )`

Table 6.9: *Reasoning rules in decision-making module*

Decision Making

The decision-making module contains reasoning rules for decision on action. The classical BDI model of 3APL [Dastani et al., 2004b] is used for the deliberation process. Table 6.9 shows the reasoning rules. The rules match the capabilities required for the firefighter role. The role description requires plans for three goals: *ExtinguishFire*, *AnswerRequest* and *InformAboutStatus*.

The agent itself has a reasoning rule for the goal to stay healthy, *TakeBreak*. Furthermore, there are two rules for the goal to extinguish a fire; if the distance is large he needs to move towards the fire, if the distance is small enough he can do the extinguish action. The fourth, fifth and sixth rule are communication rules, required for the firefighter role.

Behavior without Adjustable Autonomy

The description of the object-level reasoning specifies the standard behavior of the agent, without adjustable autonomy. The event-processing module is fixed in the order as specified in Table 6.7. We can describe the default behavior of the agent.

The agent will always follow the organizational rules. If the agent has no requests from the coordinator or other organizational goals, but it observes a fire in its environment, the agent will take initiative and extinguish the fire. If it does not observe any fires and it feels tired, it will take a rest.

We have created scenarios and tested the performance. Scenario A is a standard scenario, where occasionally a fire has to be extinguished. We have created scenarios with possibly problematic situations as well: Scenario B with high workload for the firefighter and Scenario C with wrong information at the coordinator.

In Scenario A everything looks fine, the coordination goes well. The coordinator sends commands to the agent and guides the agent to the location of the

Scenario	Duration (s)	% of fires extinguished	Lifetime firefighter
A	500	100	500
B	500	55	270
C	500	67	500

Table 6.10: *Performance without adjustable autonomy*

fire. If a new fire occurs, the same happens. In between the fires, the agent has time to take a break, and therewith, ensures its health.

The problem that occurs in Scenario B is that the firefighter does not get enough rest. The agent gets continuously requests from the coordinator. The agent does not get any rest, its fitness decreases and will in the end it gives up.

Another type of problem from organizational viewpoint is shown in Scenario C. Namely, if the coordinator has wrong location information and continuously sends requests to extinguish fires on places where there is no fire, the organization performs badly. As long as new requests reach the firefighter, it does not take initiative to extinguish fires by itself. Only when no requests are made, and it observes a fire locally, it extinguishes the fire.

The performance of the organization is measured based on the percentage of fires that is extinguished within a given period. For the firefighter we measure the time until he gives up because of tiredness. Table 6.10 shows the results for all scenarios. The performance for the standard scenario A is optimal: all fires are extinguished and the firefighter reaches the end of the scenario at time step 500. Scenario B is worst: only 55 % of the fires are extinguished and the firefighter has given up after 270 time steps because of tiredness. In scenario C, the firefighter reaches the end of the scenario, but because of the wrong information from the coordinator, he only extinguishes 67% of the fires.

6.6.2 Adjustable Autonomy

Next, we give the agent the ability of adjustable autonomy. We include metareasoning to its reasoning process. We describe the implementation of the metareasoning components below.

Performance Measure

The first part of the metareasoning process consists of monitoring and measuring the current performance. The metareasoning process analyzes the goals of the agent. When the agent is running, the goals that can occur are:

- TakeBreak

- ExtinguishFire
- AnswerRequest
- InformAboutStatus

These goals can be monitored on *urgency*, *dedication* and *confidence*. For this experiment, we describe the measures urgency and dedication. The output of the performance measure is a list of goals with corresponding values for urgency and dedication.

The urgency of a task increases over time, when a goal is active. The urgency has a value between 0 and 1. The maximum urgency is reached after 40 time steps. The following formula describes the urgency of a goal:

$$Urgency(Goal) = min(TimeActive(Goal)/40, 1)$$

The dedication of a goal depends on the origin of the goal, which can be the firefighter organization or self. We assume that the agent has chosen join the firefighter and has agreed on the organizational rules. Therefore, the dedication on the organizational goals is higher than for own goals by default. The dedication is given by a value between 0 and 1. For own goals it is fixed at 0.5. The organizational goals start at 0.7, they are dependent correct information by the coordinator. Wrong information leads to lower dedication.

```
Dedication( self ) = 0.5
Dedication( firefighter ) = 0.7
If wrong information:
    Dedication(firefighter) = max( Dedication(firefighter) - 0.2, 0)
Else:
    Dedication(firefighter) = min( Dedication(firefighter) + 0.01, 0.7)
```

Selection Mechanism

The selection mechanism creates a prioritized list of event-processing rules based on the performance measure. The meta-heuristics are:

- If the urgency of a goal exceeds UrgencyThreshold, the relevance heuristic of this goal becomes active. We define $UrgencyThreshold = 0.75$
- The priority of rules corresponds with the dedication of the goals, which is based on their origin.

The dynamics in event-processing rules that result from this process are as follows. If no attention is paid to an active goal, its urgency rises. After 30 time steps, the UrgencyThreshold is exceeded, and the meta-heuristic activates the corresponding relevance heuristic.

For example, if the agent observes tiredness, it activates the goal TakeBreak. If after 30 time steps it has not paid attention, the selection mechanism activates

ID	Origin	Rule
ep9	Self	`Observation(X) <- relevant(X, TakeBreak() ) | AddBelief(X)`
ep10	Self	`Observation(X) <- NOT relevant(X, TakeBreak() ) | IgnoreEvent()`
ep11	Self	`Message(S, P, X) <- NOT relevant(X, TakeBreak() ) | IgnoreEvent()`

Table 6.11: *Rules activated by relevance heuristic*

Priority	ID	Origin
1	ep5	Self
2	ep4	Self
3	ep6	Firefighter
4	ep7	Firefighter
5	ep8	Firefighter
6	ep1	Default
7	ep2	Default
8	ep3	Default

Table 6.12: *Prioritized list of event-processing rules after dedication towards firefighter organization decreased.*

the relevance heuristic for event processing. The rules shown in Table 6.11 are activated.

The rules ensure that only relevant observations are processed, and all other events are ignored. All other rules are deactivated.

The second type of adaptivity is that the dedication with respect to the organizational goals drops below the dedication level of its own goals. This influences the priority order of event-processing rules according to the second meta-heuristic. If this happens starting from the default situation, the effect is the prioritized set of active event-processing rules as shown in Table 6.12.

Results

If we run the same scenarios with an agent using adjustable autonomy, we expect that its results are different. First of all, the agent takes its fitness into account, and therewith takes care of itself. It will sometimes skip a request of the coordi-

Scenario	Duration (s)	% of fires extinguished	Lifetime firefighter
A	500	100	500
B	500	83	500
C	500	80	500

Table 6.13: *Results with adjustable autonomy.*

nator, in order to take a break. Furthermore, we expect that it reacts adaptively to wrong information of the coordinator. Wrong information will lead to decrease of dedication with respect to the organization, and therefore the agent will take more initiative by itself to extinguish fires.

The results of the three scenarios are presented in Table 6.13. The results support our expectations. We observe an increasing performance of the organization in both scenario B and C. In scenario B, the firefighter makes it to the end of the scenario, and manages to extinguish 83 % of the fires. In scenario C the number of fires extinguished went up from 67 % to 80 %.

With this experiment we have shown that the model for metareasoning works to achieve adaptive behavior of the agent. The coordination between coordinator and firefighter is continuously evaluated and adapted by the firefighter. The firefighter adjusts the relation by changing its event-processing rules based on the performance measure and the meta-heuristics.

We have shown two types of adaptivity. The first, via triggering the information relevance heuristic, made sure that the agent stayed alive. This was done based on the local observation about fitness. The second type of adaptivity was based on trust of other agents. It affected the organization; the agent disobeyed the coordinator and followed own observations first.

Discussion

From the experimental results we observe some relevant issues concerning the implementation of the metareasoning model. The metareasoning initiates the adaptation in the behavior of the agent. In order to make sure that the adaptation has a positive effect, the knowledge used for metareasoning should be valid. For a designer of the metareasoning process, this means that when certain knowledge is used for the metareasoning it is believed that the agent's judgment of the value of this knowledge is better than that from others.

When we translate this to the experiment described above, the agent used local knowledge to increase the urgency to take a break. It used fitness, and the fitness represents local values of the agent. The other agents in the organization could not take it into account. Therefore, the agent has the best estimation of the value of this knowledge, which makes fitness a good trigger to adapt the agent's

behavior.

The second type of adaptivity was based on trust of other agents. This is a more risky procedure. The firefighter notices locally that the coordinator fails in providing correct information. However, the firefighter can not be certain that it can make the best estimation of the coordinator's abilities. The firefighter should expect that the coordination mechanism has been designed for a certain purpose and explicitly giving priority to own goals can have negative consequences as well. In scenario B it had a good effect, but we can construct a scenario where it does not improve the organizational performance.

The conclusion to use measurements that are well judged by the agent locally is in line with architectures for emotional agents, such as [Lim et al., 2009]. Furthermore, gives interesting opportunities for learning techniques. The determination of urgency and dedication could be subject of learning, while the rest of the metareasoning process could remain the same.

6.7 Conclusion

In order to achieve optimal results, an agent's way of decision making might change according to the circumstances. To do so, an agent should be able to perform metareasoning. We argued that processing external events is a part of the reasoning process, and can be subject of metareasoning.

In this chapter we focused on controling external influences as a form of metareasoning. We combined general heuristics for event processing based on introspective, social and organizational knowledge. We allowed the agent to reason about the heuristics by applying a general model for metareasoning.

The set of active heuristics specifies the attitude of the agent towards the environment and towards other agents. The agent switches between different attitudes by activating or ignoring the heuristics. The mechanism to decide upon the best attitude is done based on measurable features of the agent's goals.

We presented a domain-independent model for reasoning about influences. We explained implementation choices and we described the adaptive behavior of an agent in a scenario involving a firefighter organization. The dynamics in the behavior follow from the feedback of the performance to a new attitude for event processing.

Chapter 7

Conclusions and Applications

This is the concluding chapter of this thesis. We summarize the results of the previous chapters and we discuss potential application areas, where the results or part of the results can be used.

7.1 Results

The motivation of this research came from a perspective on future systems where groups of actors dynamically cooridnate their actions. We studied the requirements for coordination in hybrid, dynamic organizations. These organizations consist of heterogeneous actors, who need to coordinate their actions in order to reach the organizational goals. The type of collaboration between the actors can be dynamic. In this research we have taken a technical perspective to develop a reasoning model for an artificial actor in such system. We adopted the agent paradigm as basis for the actors.

The goal of this research was to develop a reasoning model for agents capable to function in a dynamic organization. The agent should be autonomous, such that it can be held responsible for its actions, it should be able to work together with others following different types of collaboration, and it should be able to adapt the collaboration type to achieve dynamic coordination.

We argued that autonomy is a key concept for this research. In the philosophical context, autonomy is taken as basis for responsibility. We want to adopt this view to agent systems. Although autonomy is often used to define agents, there is no single definition of agent autonomy and adjustable autonomy.

7.1.1 Autonomy, Adjustable Autonomy and Coordination

In Chapter 2, we focused on autonomy and adjustable autonomy. In agent research, several approaches of autonomy and adjustable autonomy have been discussed. Some researchers describe autonomy with respect to actions and to

freedom of choice, whereas others see autonomy as a relational concept between agents. The perspective on autonomy has implications on the reasoning model of agents.

We have argued that the process of controling the internal state requires more attention in autonomy research. We have defined autonomy as *having control over external influences on the decision-making process.* Adjustable autonomy in our context means: *dynamically dealing with external influences on the decision-making process based on internal motivations.* We give the agent control over its autonomy, and therefore it is a form of self-adjustable autonomy. Adjustable autonomy gives the agent the opportunity to change its attitude with respect to the environment and to other agents. It can be used to achieve dynamic coordination.

We have presented an abstract reasoning model that facilitates agent autonomy. The two main components of the reasoning model are a component for influences control and a component for decision-making on action. We have discussed general reasoning models for reactive reasoning and BDI-reasoning in the context of autonomy.

We discussed autonomy in the context of coordination in multi-agent systems. Our view on agent autonomy guarantees the distributed feature of a multi-agent system and still allows for coordination. Autonomous actors achieve coordination by allowing influences of other agents on their decision-making process.

We discussed the relation between autonomy and existing coordination mechanisms. Coordination mechanisms have consequences for the autonomy of actors in a multi-agent system. By choosing an abstract coordination model that is defined separately from the internals of the actors, the autonomy of the actors is facilitated and the actors can be held responsible for their own actions.

7.1.2 Autonomy in the Reasoning Model

In Chapter 3, we include autonomy in the reasoning model. The two components for influence control and decision making are developed in further detail. We motivated our choice to take BDI-reasoning models as basis, because their way of semantic reasoning is intuitive and it facilitates social interaction. We extended the BDI-reasoning model with a component for processing external events. We proposed the use of reasoning rules to specify how incoming events are handled. Those event-processing rules can be used to create attitudes of the agent with respect to the outside world.

Using an experiment we showed that the agents' attitudes have an effect on organizational behavior. This shows the relation between autonomy and coordination. An extension of the experiment motivated the need for dynamic coordination mechanisms in complex environments. We demonstrated that this can be achieved in a bottom-up way by allowing the agents to change their attitude.

Finally, we presented general heuristics for influence control that can be used to process external events. Three general heuristics are based on information

relevance, organizational knowledge and trust. Having control over external influences implies that the agent chooses when to use which heuristics. This process enables adjustable autonomy.

7.1.3 Using Information Relevance for Event-Processing

In Chapter 4, we have worked out the heuristic of information relevance. Potential benefits of using information relevance for influence control can be found in controling the decision quality and preventing information overload. The benefits of relevance determination become more significant when the decision-making process makes use of complex deductions, where the deduction time depends on the size of the data base. Furthermore, the benefits of filtering information based on relevance grow as the percentage of irrelevant data gets larger.

We have proposed a way to determine the relevance of information in BDI-agents using the *Magic Sets*-method from deductive database theory. Using the magic set transformation algorithm, we construct the magic set of observations from the deduction rules in the belief base and the reasoning rules in the plan base of the agent. The agent can check the relevance of new observations based on the magic sets. We have implemented the process in 3APL agents.

7.1.4 Influence Control and Organizational Rules

Agents in an organization need to coordinate their actions in order to reach the organizational goals. Organizational models specify the desired behavior in terms of roles, relations, norms and interactions. However, the actors in an organization are autonomous entities that control their internal state and their behavior. In this research we have shown how organizational rules can be adopted by autonomous agents. We have developed a way to translate norms into event-processing rules. The event-processing rules contain a trigger, situational constraints and an effect on the agent's mental state. They are derived directly from the norms in the organizational specification.

The modular approach in our reasoning model makes possible that the event-processing rules are changed at runtime. Therewith, organizational change can be adopted by the individual agents and thus facilitates reorganization.

Since the agents are autonomous entities they will have their own event-processing rules next to the organizational rules. The modular approach makes that the agents can distinguish between different event-processing rules and that they are aware of the organizational rules. This allows for meta-reasoning about event processing, and gives the agent control over its internal state. It facilitates the autonomy of the agent and at the same time makes group coordination possible.

7.1.5 Adjustable Autonomy: Reasoning about Influence

In order to achieve optimal results, an agent's way of decision making might need to change according to the circumstances. To do so, an agent should be able to perform metareasoning. We argued that processing external events is a part of the reasoning process that can be subject of metareasoning.

In this chapter we focused on controling external influences as a form of metareasoning. We combined general heuristics for event processing based on introspective, social and organizational knowledge. We allowed the agent to reason about the heuristics by applying a general model for metareasoning.

The set of active heuristics specifies the attitude of the agent towards the environment and towards other agents. The agent switches between different attitudes by activating or ignoring the heuristics. The mechanism to decide upon the best attitude is done based on measurable features of the agent's goals.

We presented a domain-independent model for reasoning about influences. We explained implementation choices and we described the adaptive behavior of an agent in a scenario involving a firefighter organization. The dynamics in the behavior follow from the feedback of the performance to the module for event-processing. With the ability to control external influences based on current performance the agent shows self-adjustable autonomy.

7.1.6 Main Results

The goal of this research was to develop a reasoning model for agents capable to function in a dynamic organization. The agent should be autonomous, such that it can be held responsible for its actions, it should be able to work together with others following different types of coordination, and it should be able to adapt the coordination type to achieve dynamic coordination. In the previous chapters we have described the search for such model.

In order to achieve our goal, we have taken a new perspective on the relation between autonomy and coordination. The approach that we have taken, started from the assumption that the agents autonomous. We have argued that an autonomous agent should control external influences on its decision-making process. By making influence control a dynamic process, the agent can adjust its autonomy relative to others. Coordination between autonomous actors, then, can be achieved by explicitly allowing influences on the decision-making process. This way of relating autonomy and coordination is new in agent research. We believe it is a powerful basis to design autonomous actors for dynamic organizations, and to model organizational performance.

We have identified two reasoning components, that enable agents for be autonomous: a component for influence control and a component for decision making. Although this is an abstract description of reasoning, it points out crucial processes concerning autonomy and coordination, that should be dealt with when designing the agents. Both are local processes.

We have presented a possible implementation this abstract reasoning model, where we have chosen for a modular approach to implement the reasoning components. We used BDI-reasoning for the decision-making component and rule-based event-processing for influence control. We stress that this is not the only possible implementation. However, we want to point out some benefits of our approach.

- The modularity in our reasoning model separates event processing and decision making. This allows the development of domain-independent heuristics for event-processing. In this thesis we have mentioned three of them: information relevance, organizational knowledge and trust. The first two have been developed in further detail in this thesis.
- The modularity in our reasoning model ensures that the event-processing rules are explicitly defined. This allows for metareasoning about the event-processing rules. Therewith, the agent can determine the desired attitude with respect to the environment and to other agents, and it can modify the event-processing rules to take up the desired attitude. This process gives the agent control over external influences, and makes that it meets the requirements for autonomy and adjustable autonomy.

By fulfilling the requirements of autonomy and adjustable autonomy we have developed a model for agent reasoning, that makes the actors capable to function in a dynamic organization, where they are responsible for their actions and where they are able to follow dynamic coordination mechanisms, and can take initiative to adapt the coordination.

7.2 Towards Applications

We have presented a view on agent autonomy, adjustable autonomy and the relation with coordination. We have implemented prototypes to demonstrate and support the theories and we have done experimentations within limited simulation environments. More than a software implementation, the main contribution of this research is a new view on autonomy and coordination that can be useful when designing complex systems. In this section we discuss several application areas where the results of our research have been used or can be used. This study was part of the Interactive Collaborative Information Systems (ICIS) project [, ICIS]. All application areas presented here, are directly connected to the ICIS problem domain.

7.2.1 Human-Agent Teams

The first type of applications we discuss is based on the original research goals to investigate dynamic task division between humans and systems. We describe adaptive support systems, where an agent-based system gives dynamic support to a human operator. Furthermore, we discuss development of augmented teams.

Adaptive Support Systems

Adaptive Support Systems was one of the starting topics of this research. As we talk about human support, the system needs to interact with a user and, therewith, human-computer interaction is a related research area. Although we have not actually made implementations and experimentations in a human computer interaction, the findings in this research can contribute to system design. We have presented a model to achieve adaptive coordination in autonomous systems. Some of our results have been used to specify the adaptivity in human support systems, as discussed by Neef [Neef et al., 2009a].

Based on studies on human-aware agents [Maanen et al., 2008] and on work-centered support systems [Scott et al., 2005], Neef et al. propose a support system that targets the attention allocation problem and task overload issue for tactical officers [Neef et al., 2009a]. Maanen [Maanen et al., 2008] describes a human-aware mechanism for assisting humans in appropriately allocating their attention. This mechanism has been evaluated in the domains of air traffic control, computer games, and in tactical picture compilation.

The extended model combines both human-aware and work-centered support. Neef et al. combine the mechanism for attention allocation with the possibility of adjusting the agreements on task division between the human operator and the support agent. They specify the boundaries for the system adaptivity and the current task-division rules using the social contracts of the OperA framework [Dignum, 2004]. The adaptive behavior of the agents is achieved via the theory of influence control presented in this thesis.

The support agent adopts the rules for task division into the influence control module. The agents reason about the task division based on the current performance. They are able to modify the task division within the specified boundaries by changing their rules for influence control. This way, the system gives the responsibility for adaptive behavior to the agent, while giving the operator straightforward means to specify the boundaries of adaptivity.

Augmented Teams

In other recent work Neef et al. [Neef et al., 2009b] study the development of *augmented teams*: human teams whose capabilities are augmented by the involvement of sensors, networks and artificial actors. The resulting system consists of human actors and artificial actors. The technological means become part of the human team itself. This requires adaptive capabilities from the artificial actors in the system.

Development of such systems, where human actors and the artificial actors are considered as part of the system, is studied in cognitive systems engineering [Klein et al., 2004]. Neef et al. [Neef et al., 2009b] present a model to specify cognitive systems. They propose to describe task division in a similar manner as we have done in Chapter 5 using OperA: via role descriptions and interaction

contracts. Therewith, behavioral guidelines are specified, but the actos stay fully autonomous in determining their actions. The artificial actors should be able to function in a dynamic coordination mechanism and take initiative for organizational change. The results of our work can be used for the implementation of the artificial actors. The adaptivity of the actors can be achieved in the way we discussed in Chapters 5 and 6, where the agents reason about the organizational rules and are able to adjust them locally.

7.2.2 Modeling Organizations and Coordination Mechanisms

When modeling organizational processes it is important to take both the organizational perspective as the individual into account. In this thesis we have presented a new perspective on the relation between agent autonomy and group coordination. We have shown how to use agent-based modeling for the implementation of the individual behavior of participants in an organization. We have adopted the OperA approach to design and specify the framework of coordination. The individual actors can adopt organizational rules in a reasoning module for event-processing. Therewith, we presented a method to model organizations using both organizational and individual perspective.

Network Enabled Capabilities

In the military domain there is a lot of research on effectiveness of military organizations under mission critical circumstances. Research on Command and Control (C2) is generally done from an organizational perspective. There is research on organizational structures and the effect on human factors [SAS026, 2002].

Dekker [Dekker, 2005] has made a taxonomy of C2 coordination models that follow from possible network structures. He distinguishes two dimensions: homogeneity/heterogeneity of the participants and value-symmetry/non-value-symmetry of the roles they perform. From these dimensions he creates eight different coordination models. The three basic types are: centralized control, request-based coordination and emergent coordination. The other five are combinations of the basic types.

Network Enabled Capabilities (NEC) is a popular future perspective on C2 organizations. The idea is that every actor is considered as a node and all nodes are part of a networked organization. The network structure of an organization determines the information flow within the organization [Alberts and Hayes, 2003]. The philosophy behind the networked structure is that organizations become flexible in assigning roles and tasks to actors.

Organizational simulation in the NEC community is mainly done at an abstract level, for example, based on social simulations. Dekker [Dekker, 2006] analyzes the network structure via the information flow within the network. However, no tasks are actually executed. We believe that simulations can be made more powerful when the individual behavior of the actors is implemented as well.

Then, the organizational performance is dependent of both individual behavior and organizational structure.

The taxonomy of coordination types [Dekker, 2005] can perfectly be specified using Opera. We have shown how autonomous agents can adopt the organizational rules. We can include individual behavior in the organizational simulations. Therewith, the results from this thesis can contribute to research on C2 and NEC simulations.

7.2.3 Information Relevance in Robots, Games and Teams

In Chapter 4 we discussed information processing based on its relevance. We have presented the heuristic in the context of influence control. Here, we show that awareness of information relevance has a larger use.

First of all, information relevance is an important notion when an agent wants to function autonomous with respect to its environment. This becomes an issue when an agent is connected to an external environment. The designer of the agent does not control the environment, and thus does not control the external events that the agent will experience. Assume that we connect a 3APL agent [Dastani et al., 2004b] to an external game engine, in order to program the characters in the game. The game engine sends updates of the complete world state several times per second. The information load is too big to handle in the decision-making module. Moreover, only part of the information will be used in the agent's reasoning process. If we have a notion of relevance, we can create a filter on the input received from the game engine.

The same holds when connecting a reasoning module to a robot platform. The robot can have many different sensors that often continuously do measurements and provide new information to the reasoning module. It is very likely that depending on the tasks only part of the sensor information is useful for the reasoning process. Therefore, a filter on the input can save a lot of information-processing capacity for the reasoning model. The filter should be connected to the decision-making process in order to use the current goals. This results in adaptive influence control in a similar manner as we have presented.

From the examples, it becomes clear that relevance determination needs to be a local process, as it depends on local goals and tasks. However, within organizations and coordination mechanisms information relevance plays an important role as well. If it is possible to determine which information is relevant to another actor, it becomes more easy to get the right information to the right place at the right time.

In Chapter 4, we have presented a way to specify relevance of information for agents that use logical reasoning. The relevance definition is based on magic-set theory [Beeri and Ramakrishnan, 1991]. This theory comes from deductive database research, where a lot of study has been done to deal efficiently with large amounts of information. We have made an attempt to transfer this knowledge to agent reasoning, we presented an algorithm to determine relevance and we

achieved a practical implementation. Depending on the applications the magic-set approach will be more or less beneficial. However, we believe that future research in this area could provide solutions for the mentioned applications.

7.3 Concluding Thoughts

In this dissertation we have presented a model for agent reasoning, that makes agents capable to function in a dynamic organization, where they are responsible for their actions and where they are able to follow dynamic coordination mechanisms, and can take initiative to adapt the coordination.

We have presented a view on the relation between autonomy of agents and coordination of group behavior. An autonomous agent should control external influences on its decision-making process. By making influence control a dynamic process, the agent can adjust its autonomy relative to others. Coordination between autonomous actors, then, can be achieved by explicitly allowing influences on the decision-making process.

One can think of many future applications where the topics issued in this thesis are relevant. We have taken a domain-independent approach and we believe that the resulting concepts are useful for practical system design. However, when designing real-world applications many domain dependent choices have to be made that determine the success or failure of the system. Therefore, every future research project on human-agent teams and dynamic coordination will experience interesting challenges.

Appendix A

Magic Sets Transformation

The magic set method is a bottom-up query evaluation technique first introduced in [Bancilhon et al., 1986]. A straight forward algorithm for the Magic Set transformation is explained in [Beeri and Ramakrishnan, 1991]. Magic Sets are used to define the relevant predicates in a database for a specific query in order to speed up the search process significantly.

Consider a deductive database D that consists of relations (facts), and a program P (derivation rules). A query Q is written as $q(c, X)$, where some variables of the query are bound (c) and others are open (X) and need to be derived. A query-program pair (P, Q) and a database D result in a set of bindings for variables X that make the query expression true.

The Magic Set method evaluates program P in a top-down manner, using the variable bindings from the query Q. The program P is rewritten using sideways information passing in the derivation rules. The result is a new program P^{ad}, that is equivalent to P with respect to the query.

Next, we define magic sets for all derivable predicates, in which we store the allowed values of variable bindings for this predicate. By adding these magic predicates to the derivation rules, the possible derivations are restricted by the magic set. Any evaluation strategy applied to the magic program will restrict all predicate derivations to those that are relevant according to the top-down strategy.

In database research, the Magic Set transformation is used to rewrite a program and a query to a more efficient program for the same query. The new program searches the database in a much more efficient way. The method is complete, such that we do not throw any relevant information away [Beeri and Ramakrishnan, 1991]. It is especially useful when recursion occurs in the program, because the bounded elements from the query have been evaluated in the derivation rules already. However, it also shows significant benefits in relational databases without recursion [Mumick and Pirahesh, 1994].

A.1 Magic Sets Transformation Algorithm

The algorithm to create a magic set for a program P and query Q contains the following steps [Beeri and Ramakrishnan, 1991].

1. Create the adorned rule set P^{ad} from P and Q using sideways information passing a top-down evaluation strategy.
2. Create a new predicate $magic\_p^a$ for each p^a in P^{ad} (thus we create magic predicates only for derived predicates in P^{ad}). The arity of the new predicate is the number of occurrences of b in the adornment a, and its arguments correspond to the bound arguments of p^a.
3. For each rule r in P^{ad} and for each occurrence of an adorned predicate p^a in its body, we generate a magic rule defining $magic\_p^a$. (Note that an adorned predicate may have several occurrences, even in one rule, so several rules that define $magic\_p^a$ may be generated from a single adorned rule)
4. Each rule is modified by the addition of magic predicates to its body.
5. Create a seed $magic_q(c)$ for the magic predicates, in the form of a fact, obtained from the query $q(c, X)$, where c are the bound variables in the query and X are the free variables.

A.2 Example

Consider the example we used before, describing *ancestor* relations based on *parent* facts. We describe program P, query Q and database D:

```
P:
r1    anc(X,Y) :- par(X,Y)
r2    anc(X,Y) :- par(X,Z), anc(Z,Y)

Q:    anc(john,Y)?

D:   database containing a set of par relations.
```

Using a sideway-information passing strategy the variable bindings are taken to following evaluations. The query Q has some variables bound (b) and some free (f). Given a program P and a query Q, we can rewrite P to the adorned rules P' in which the variable bindings indicated for every predicate that needs to be derived. In our example, the query Q is $anc(john, Y)$, which has bindings bf:

```
P':
r1'   anc_bf(X,Y) :- par(X,Y)
r2'   anc_bf(X,Y) :- par(X,Z), anc_bf(Z,Y)

Q:    anc_bf(john,Y)?
```

The adorned rules are created by evaluating the predicates from right to left, and remembering the derived variable bindings. The *par* relation is a fact that is taken straight from the database. Therefore, no bindings need to be given. In rule $r2$, first $par(X, Z)$ is evaluated, which will bind variable Z. So $anc(Z, Y)$ has adornment bf.

Looking at the example, we want to restrict the computation only to the ancestors of *john*. A new predicate is defined, in which we intend to store the values for which anc needs to be computed. This predicate is called the *magic* predicate:

```
magic_anc_bf(john)
```

From rule $r2'$ of P' we can conclude that if we need to compute the ancestors of X, and par(X,Z) holds, we also need to compute the ancestors of Z. Therefore, we can generalize the magic predicate $magic\_anc\_bf$ to:

```
magic_anc_bf(Z) :- magic_anc_bf(X), par(X,Z)
```

The result is that we have defined which set of $anc\_bf(X, Y)$ need to be evaluated based on query Q. We have to restrict general evaluation rules of $anc\_bf(X, Y)$ such that only relevant predicates will be evaluated. Therefore we add the magic predicate to the derivation rules from P'. The complete Magic Program Pm, derived from P and Q contains the modified rules $r1'$ and $r2'$ together with the magic predicates.

```
Pm:
r1'   anc_bf(X,Y) :- magic_anc_bf(X), par(X,Y)
r2'   anc_bf(X,Y) :- magic_anc_bf(X), par(X,Z), anc_bf(Z,Y)
r3'   magic_anc_bf(Z) :- magic_anc_bf(X), par(X,Z)
r4'   magic_anc_bf(john)
```

This magic program Pm leads to the same answers for the query, but is more efficient way, such that no irrelevant predicates are evaluated.

Bibliography

D. S. Alberts and R. E. Hayes. *Power to the Edge*. CCRP Publication Series, 2003.

F. Bancilhon, D. Maier, Y. Sagiv, and U. J. Magic sets and other strange ways to implement logic programs. *Proc. 5th Symposium on Principles of Database Systems*, 1986.

K. S. Barber, D. C. Han, and T.-H. Lui. Strategy selection-based meta-level reasoning for multi-agent problem-solving. *Proceedings of the Autonomy Control Software Workshop at AA-1999*, pages 3247–3259, 2000.

K. S. Barber and J. Kim. Belief revision process based on trust: Agents evaluating reputation of information sources. In *Trust in Cyber-societies*, pages 73–82. 2001.

C. Beeri and R. Ramakrishnan. On the power of magic. *Journal of Logic Programming*, 10:pages 255–300, 1991.

R. Bordini, J. Hübner, and R. Vieira. Jason and the golden fleece of agent-oriented programming. In R. Borini, M. Dastani, J. Dix, and A. E. F. Seghrouchni, editors, *Multi-Agent Programming; Languages, Platforms and Applications*, pages 3–37. Springer, 2005.

T. Bosse, C. Jonker, J. Treur, and D. Tykhonov. Formal analysis of trust dynamics in human and software agent experiments. In M. Klusch, H. Hindriks, P. M.P., and L. Sterling, editors, *Cooperative Information Agents XI, Proceedings of the Eleventh International Workshop on Cooperative Information Agents, CIA'07*, volume 4676 of *Lecture Notes in Artificial Intelligence*, pages 343–359. Springer Verlag, 2007.

J. M. Bradshaw, P. J. Feltovich, H. Jung, S. Kulkarni, W. Taysom, and A. Uszok. Dimensions of adjustable autonomy and mixed-initiative interaction. In M.Nickles, M. Rovatsos, and G. Weiss, editors, *Autonomy 2003*, volume 2969 of *LNAI*, pages 17–39. 2004.

J. M. Bradshaw, H. Jung, S. Kulkarni, M. Johnson, P. Feltovich, J. Allen, L. Bunch, N. Chambers, L. Galescu, R. Jeffers, N. Suri, W. Taysom, and A. Uszok. Kaa: policy-based explorations of a richer model for adjustable autonomy. In *AAMAS '05: Proceedings of the fourth International Conference on Autonomous Agents and Multiagent Systems*, pages 214–221. ACM, New York, NY, USA, 2005.

V. Braitenberg. *Vehicles: Experiments in synthetic psychology*. MIT Press, Cambridge, MA, USA, 1984.

M. E. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, USA, 1987.

G. Brewka. Reasoning about priorities in default logic. In *AAAI'94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 2)*, pages 940–945. American Association for Artificial Intelligence, 1994.

C. Carabelea, O. Boissier, and A. Florea. Autonomy in multi-agent systems: A classification attempt. *Agents and Computational Autonomy: Potentials, Risks and Solutions*, LNAI 2969:pages 103–113, 2004.

C. Castelfranchi. Guarantees for autonomy in cognitive agent architecture. In *Intelligent Agents*, volume 890 of *Lecture Notes in Artificial Intelligence*, pages 56–70. Springer Berlin / Heidelberg, 1995.

C. da Costa Pereira, T. A., and L. Amgoud. Goal revision for a rational agent. *Proceedings of the 17th European Conference on Artificial Intelligence, ECAI06*, pages 747–748, 2006.

M. Cox. Metacognition in computation. *Artificial Intelligence*, 169(2):pages 104–141, 2005.

M. Cox and A. Raja. Metareasoning: A manifesto. *Proceedings of Metareasoning: Thinking about Thinking*, 2008.

M. Dastani. The cyclic interpreter (deliberation cycle) for 3apl agents. http://www.cs.uu.nl/3apl/deliberationcycle.pdf.

M. Dastani, F. Dignum, and J.-J. C. Meyer. Autonomy and agent deliberation. *Agents and Computational Autonomy: Potentials, Risks and Solutions*, LNAI 2969:pages 114–127, 2004a.

M. Dastani, V. Dignum, and F. Dignum. Role assignment in open agent societies. In *AAMAS03*, pages 489 – 496. ACM Press, 2003.

M. Dastani, B. van Riemsdijk, F. Dignum, and J.-J. C. Meyer. A programming language for cognitive agents: Goal directed 3apl. *ProMAS'03*, LNAI 3067:pages 111–130, 2004b.

A. Dekker. A taxonomy of network centric warfare architectures. In *SETE 2005 Systems Engineering, Test and Evaluation Conference*. Brisbane, Australia, 2005.

A. Dekker. Agility in networked military systems: A simulation experiment. In *11th International Command and Control Research and Technology Symposium (ICCRTS)*. Cambridge, UK, 2006.

D. Dennett. *The Intentional Stance*. MIT Press, Cambridge, Mass, 1987.

J. van Diggelen, R.-J. Beun, R. M. van Eijk, and P. J. Werkhoven. Agent communication in ubiquitous computing: the ubismart approach. In Padgham, Parkes, Müller, and Parsons, editors, *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 813–820. 2008.

F. Dignum, D. Kinny, and L. Sonenberg. From desires, obligations and norms to goals. *Cognitive Science Quarterly*, 2(3-4):pages 407–430, 2002.

V. Dignum. *A Model for Organizational Interaction: based on Agents,founded in Logic*. Utrecht University, PhD Thesis, 2004.

V. Dignum, F. Dignum, and J.-J. C. Meyer. An agent-mediated approach to the support of knowledge sharing in organizations. *Knowledge Engineering Review*, 19(2):pages 147–174, 2004a.

V. Dignum, J.-J. C. Meyer, F. Dignum, and H. Weigand. Formal specification of interaction in agent societies. In M. Hinchey, J. Rash, W. Truszkowski, C. Rouff, and D. Gordon-Spears, editors, *Formal Approaches to Agent-Based Systems (FAABS'02)*, volume 2699 of *LNAI*, pages 37–52. Springer, 2003.

V. Dignum, L. Sonenberg, and F. Dignum. Dynamic reorganization of agent societies. In *Proceedings of CEAS: Workshop on Coordination in Emergent Agent Societies at ECAI 2004*, pages 42–55. Valencia, Spain, 2004b.

M. d'Inverno and M. Luck. Engineering agentspeak(l): A formal computational model. *Journal of Logic and Computation*, 1998.

G. Dworkin. *The Theory and Practice of Autonomy*. Cambridge University Press, New York, 1988.

R. Elmasri and S. B. Navathe. *Fundamentals of database systems (2nd ed.)*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1994.

M. D. Eric Bonabeau and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

M. Esteva, J. Padget, and C. Sierra. Formalizing a language for institutions and norms. In J.-J. C. Meyer and M. Tambe, editors, *Intelligent Agents VIII*, volume 2333 of *LNAI*, pages 348–366. Springer, 2001.

R. Falcone and C. Castelfranchi. The human in the loop of a delegated agent: the theory of adjustable social autonomy. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 31(5):pages 406–418, 2001.

J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings of the 3rd International Conference on Multi Agent Systems*, page 128. IEEE Computer Society, 1998.

A. Garca-Camino, J. A. Rodrguez-Aguilar, and W. W. Vasconcelos. A distributed architecture for norm management in multi-agent systems. In *Coordination, Organization, Institutions and Norms in Agent Systems III*, volume 4870 of *LNAI*, pages 275–286. Springer, 2007.

M. Ghijsen, W. Jansweijer, and B. Wielinga. Towards a framework for agent coordination and reorganization, agentcore. In J. S. Sichman, J. Padget, S. Ossowski, and P. Noriega, editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, volume 4870 of *LNCS*, pages 1–14. Springer-Verlag Berlin Heidelberg, 2008.

E. A. Hansen and S. Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, 126(1-2):pages 139–157, 2001.

K. V. Hindriks, F. S. D. Boer, W. V. der Hoek, and J.-J. C. Meyer. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):pages 357–401, 1999.

J. H. Holland. *Emergence. From Chaos to Order*. Addison-Wesley, New York, NY, USA, 1998.

J. Hübner, J. S. Sichman, and O. Boissier. Moise+: towards a structural, functional, and deontic model for mas organization. *Proceedings of AAMAS02*, pages 501–502, 2002.

J. Hübner, J. S. Sichman, and O. Boissier. Using the moise+ for a cooperative framework of mas reorganisation. In *Advances in Artificial Intelligence SBIA 2004*, volume 3171 of *LNCS*, pages 505–515. Springer, 2004.

I. C. I. S. (ICIS). The website of the icis project. http://www.icis.decis.nl/.

N. R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):pages 277–296, 2000.

A. Kakas, N. Maudet, and P. Moraitis. Modular representation of agent interaction rules through argumentation. *Journal of Autonomous Agents and Multi-Agent Systems*, 11(2):pages 189–206, 2005.

I. Kant. *Grundlegung zur Metaphysik der Sitten (English: Groundwork of the Metaphysics of Morals)*. 1785.

G. Klein, D. Woods, J. Bradshaw, R. Hoffman, and P. Feltovich. Ten challenges for making automation a "team player" in joint human-agent activity. *IEEE Intelligent Systems*, 19(6):pages 91–95, 2004.

M. Y. Lim, J. Dias, R. Aylett, and A. Paiva. Intelligent npcs for education role play game. In *To be published at Workshop on Adaptive Agents for Games, AAG@AAMAS2009*. 2009.

P.-P. v. Maanen, L. d. Koning, and K. v. Dongen. Design and validation of habta: Human attention-based task allocator. In M. Mhlhuser, A. Ferscha, and E. Aitenbichler, editors, *Constructing Ambient Intelligence: AmI-07 Workshops Proceedings*, volume 11 of *Communications in Computer and Information Science (CCIS)*, pages 286–300. Springer Verlag, 2008.

C. E. Martin and K. S. Barber. Adaptive decision-making frameworks for dynamic multi-agent organizational change. *Autonomous Agents and Multi-Agent Systems*, 13(3):pages 391–428, 2006.

E. Matson and S. DeLoach. Formal transition in agent organizations. In *IEEE International Conference on Knowledge Intensive Multiagent Systems (KIMAS '05)*, pages 235–240. 2005.

M. McCallum, W. W. Vasconcelos, and T. J. Norman. Organisational change through influence. *Journal of Autonomous Agents and Multi-Agent Systems*, 2008.

I. S. Mumick and H. Pirahesh. Implementation of magic-sets in a relational database system. In *Proceedings of the 1994 ACM SIGMOD international conference*, pages 103–114. 1994.

R. Neal. *Bayesian Learning for Neural Networks*. Springer, Heidelberg, 1996.

M. Neef. A taxonomy of human - agent team collaborations. In *Proceedings of the 18th BeNeLux Conference on Artificial (BNAIC 2006)*, pages 245–250. 2006.

M. Neef, P.-P. van Maanen, P. Petiet, and M. Spoelstra. Adaptive work-centered and human-aware support agents for augmented cognition in tactical environments. In *Accepted for the HCI International - Augmented Cognition 2009*. 2009a.

M. Neef, M. van Rijn, D. Keus, and J.-W. Marck. Organizing smart networks and humans into augmented teams. In *Accepted for the HCI International - Augmented Cognition 2009*. 2009b.

N. Oren, M. Luck, S. Miles, and T. J. Norman. An argumentation inspired heuristic for resolving normative conflict. In *Proceedings of the Workshop on Coordination, Organization, Institutions and Norms in Multi-Agent Systems 2008*. 2008.

A. S. Rao. Agentspeak(l): Bdi agents speak out in a logical computable language. In W. V. de Velde and J. Perram, editors, *Proceedings of the Seventh Workshop on Modelling Autonomous Agents in a Multi-Agent World*, volume 1038 of *LNAI*, pages 42–55. Springer, 1996.

A. S. Rao and M. P. Georgeff. Bdi-agents: from theory to practice. In *Proceedings of the First ICMAS*. San Francisco, 1995.

M. B. v. Riemsdijk, F. S. d. Boer, and J.-J. C. Meyer. Dynamic logic for plan revision in intelligent agents. *Proceedings of the fifth international workshop on computational logic in multi-agent systems (CLIMA'04)*, LNAI 3487:pages 16–32, 2005.

S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (2nd ed.)*. Prentice Hall, Upper Saddle River, NJ, USA, 2003.

N. SAS026. *The NATO Code of Best Practice for C2 Assessment*. CCRP Publication Series, 2002.

P. Scerri, K. Sycara, and M. Tambe. Adjustable autonomy in the context of coordination. *AIAA 3rd "Unmanned Unlimited"*, 2004.

R. Scott, E. M. Roth, S. E. Deutsch, E. Malchiodi, T. Kazmierczak, R. Eggleston, S. R. Kuper, and R. Whitaker. Work-centered support systems: A human-centered approach to intelligent system design. *IEEE Intelligent Systems*, 2(20):pages 73–81, 2005.

J. Sichman and R. Conte. On personal and role mental attitude: A preliminary dependency-based analysis. In *Advances in AI*, LNAI 1515, pages 1–10. Springer, 1998.

D. Sperber and D. Wilson. *Relevance: communication and cognition*. Harvard University Press, Cambridge, MA, USA, 1986.

M. Tambe, P. Scerri, and D. Pynadath. Adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, (17):pages 171–228, 2002.

B. v. d. Vecht, F. Dignum, and J.-J. C. Meyer. Magic agents: Using information relevance to control autonomy. In M. Ghallab, C. D. Spyropoulos, N. Fakotakis, and N. Avouris, editors, *Proceedings of the ECAI 2008 - 18th European Conference on Artificial Intelligence*, pages 889–890. 2008a.

B. v. d. Vecht, F. Dignum, and J.-J. C. Meyer. Autonomous agents adopting organizational rules. In V. Dignum, editor, *Multi-Agent Systems: Semantics and Dynamics of Organizational Models, Chapter 13*, pages 314–333. IGI Global Press, 2009.

B. v. d. Vecht, A. P. Meyer, R. M. Neef, F. Dignum, and J.-J. C. Meyer. Influence-based autonomy levels in agent decision-making. In P. Noriega, J. Vzquez-Salceda, G. Boella, O. Boissier, V. Dignum, N. Fornara, and E. Matson, editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, volume 4386 of *Lecture Notes in Computer Science*, pages 322–337. Springer-Verlag, Berlin/Heidelberg, 2007.

B. v. d. Vecht, J.-J. C. Meyer, F. Dignum, and M. Neef. A dynamic coordination mechanism using adjustable autonomy. In J. Sichman, P. Noriega, J. Padget, and S. Ossowski, editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, volume 4870 of *Lecture Notes in Computer Science*, pages 83–96. Springer-Verlag, Berlin/Heidelberg, 2008b.

H. Verhagen. *Norm Autonomous Agents*. Stockholm University, PhD Thesis, 2000.

M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):pages 115–152, 1995.

Summary

As result of technological developments we foresee future systems where groups of actors coordinate their actions in a dynamic manner to reach their goals. In human-machine interaction the machine will act more as a team member instead of a user tool and in distributed systems artificial actors will cooperate autonomously. For example, consider a future traffic situation where cars take over part of the driver's actions in accelerating, breaking and steering. Cars might be able to communicate with other vehicles to exchange information and might be able to adapt their speed to one another. The cars in such system are able to take initiatives and have certain responsibilities with respect to their actions. Other types of applications are, for example, mixed human-robot teams, or sensor networks where sensors are embedded in entities that can perform actions and exchange information, or gaming and simulations with advanced virtual characters.

Our aim is to develop a reasoning model for artificial actors in such systems. The actors should be autonomous such that they make their own decisions and they can be held responsible for their actions. Furthermore, they should be able to work together following different types of coordination in order to achieve dynamic coordination.

Starting point of our research is the relation between autonomy of individuals and coordination of group behavior. An autonomous actor determines its own actions, whereas coordination requires tuning of activities to one another. We adopt the agent paradigm as basis for the actors. Agents are considered to be entities with reasoning capabilities, that are able to perform actions. Although autonomy is a key concept in agent research, there is no common definition of agent autonomy and adjustable autonomy.

In this research, we define being autonomous as *having control over external influences on the decision-making process.* This means that the agent determines to what extend its decisions are influenced by the environment and by other agents. In the opposite situation, in which the agent has no control over influences, its decisions is fully depend on the environment and the agent can be manipulated by others. We consider influence control as an adaptive process, controlled by the agent itself. Therewith, autonomy becomes an adjustable feature. Adjustable autonomy in our context means *dynamically dealing with external influences on the decision-making process based on internal motivations.* This implies that agents

can choose to be open to certain influences. Because the agents can adapt their openness to one another, they can achieve coordination by allowing influences on their decision-making process.

This perspective on autonomy puts requirements on the internal structure of an agent. In Chapter 3 of this thesis we present an abstract reasoning model that facilitates agent autonomy and we put forward an implementation specification. The two main components of the reasoning model are *influence control* and *decision making*. In the implementation we have used the Belief-Desire-Intention (BDI-)model for the decision-making process. A BDI-agent has an internal state containing beliefs, goals and plans. Based on the internal state it decides upon the desired actions.

In the component for influence control the agent determines to what extend its internal state is influenced by external events. Here, the agent processes new observations, inform messages and request messages based on considerations as: is this information relevant for me? Should I accept this request? Should I believe this message? We propose to use reasoning rules to process those external events. The reasoning rules specify the effects of external events on the beliefs and goals of the agent given certain constraints. The beliefs and goals, then, are used for the decision-making process.

With this two-piece reasoning model we do a coordination experiment. In a simulated environment a firefighter organization needs to extinguish several fires. The organizational goal can be reached by allocating the firefighters to the fires. This allocation process can be done via different types of coordination ranging from *emergent coordination* to *centralized coordination*. The experiment shows that the perspective on autonomy as influence control provides a way to achieve different types of coordination. Also, adjustable autonomy seems a promising way to facilitate dynamic coordination. However, some questions arise when giving individual agents control over external influences. First of all, the agent should process the events in a way that makes sense. What are sensible heuristics for event-processing? Furthermore, the agent should adjust its attitude towards others and towards the environment based on the situation. How do we obtain this adaptivity?

We can tackle both aspects by exploiting the *modularity* of the reasoning model. Our reasoning model separates event-processing and decision-making. This separation allows for the development of domain-independent heuristics for event-processing. In this thesis we mention three heuristics: information relevance, organizational knowledge and trust. The first two are discussed in further detail.

In Chapter 4, we discuss potential benefits of using information relevance for influence control. The intuition is that not all information is relevant for all tasks. For example, when driving a car from A to B in one side of the country, traffic information about the other side of the country is not relevant to achieve your goal. Using information relevance for influence control allows the agent to focus on a specific goal and prevents information overload. In this chapter, we propose

an algorithm for relevance determination in BDI-agents based on the *magic sets*-method, which has been developed for efficiently searching deductive databases. We have adapted the method slightly to apply to BDI-reasoning.

We work out the use of organizational knowledge to process external events in Chapter 5. When becoming part of an organization one agrees to certain behavioral rules. For example, you should stop for a red light, or you should inform your boss when you are late. In this chapter, we show how organizational norms can be translated into event-processing rules. New observations or messages can trigger obligations or prohibitions. Furthermore, our representation of the event-processing rules facilitates organizational dynamics. The event-processing rules can be changed at any time. Therewith, changes within the organization can be adopted by the individual agents.

In Chapter 6, we focus on adjustable autonomy. We extend our reasoning to allow the agent to reason about the heuristics for influence control: Are the organizational norms valuable? How to deal with contradicting norms? Is relevance an issue when processing observations? The modularity in our reasoning model ensures that the event-processing rules are explicitly defined. This allows for metareasoning about the event-processing rules. We present a metareasoning model, with which the agent can select and take up the desired attitude with respect to the environment and to other agents. For instance, it can choose to focus on a goal when a deadline is approaching, or it can choose to follow the organizational norms. With this process the agent gets control over external influences, and therewith it meets the requirements for autonomy and adjustable autonomy.

In a simulation experiment we implement a firefighter organization again and we show that the organization exhibits dynamic coordination via self-adjustable autonomy of the firefighters. With this experiment we demonstrate that individuals contribute to the degree of adaptivity of the organization by controling to what extend they are being influenced. The artificial actors in our experiment can deal with different types of cooperation and can take initiative to adapt the coordination type. Therewith, we create the opportunity to develop systems in which human beings as well as artificial actors coordinate in a dynamic manner to achieve their goals.

Samenvatting

Als gevolg van technologische ontwikkelingen zullen er in de toekomst systemen ontstaan waarin menselijke en kunstmatige actoren op een dynamische manier hun acties coördineren om hun doelen te bereiken. In mens-machine-interactie zullen machines meer als teamgenoot functioneren dan als gebruiksvoorwerp en in gedistribueerde systemen zullen kunstmatige actoren onderling samenwerken. Denk bijvoorbeeld aan het verkeer in de toekomst. Auto's zullen taken als gas geven, remmen en sturen overnemen van de bestuurder. Ze zullen communiceren met andere voertuigen om informatie uit te wisselen en ze zullen hun snelheid aan elkaar aanpassen. In een dergelijk systeem kunnen de auto's zelf initiatief nemen en zijn ze tot op zekere hoogte zelf verantwoordelijk voor hun acties. Andere voobeelden van toepassingsgebieden zijn mens-robotteams, of sensornetwerken waarin de sensoren acties kunnen uitvoeren en informatie kunnen uitwisselen, of gaming- en simulatietoepassingen met geavanceerde virtuele personages.

Het doel van dit onderzoek is om een redeneermodel te ontwikkelen voor kunstmatige actoren in een dergelijk systeem. De actoren moeten autonoom zijn, zodat ze zelf beslissen wat ze doen en verantwoordelijk kunnen worden gehouden voor hun eigen acties. Verder moeten ze op verschillende manieren kunnen samenwerken om dynamische vormen van coördinatie te bereiken.

We beginnen met het bestuderen van de relatie tussen autonomie van een individu en coördinatie van groepsgedrag. Een autonome actor bepaalt zelf wat hij doet, terwijl bij cöordinatie onderlinge afstemming vereist is. Als basis voor de kunstmatige actoren gebruiken we agenttechnologie. Agenten zijn entiteiten met het vermogen om te redeneren en om zelfstandig acties uit te voeren. Alhoewel autonomie een belangrijk concept is in onderzoek over agenttechnologie, is er geen algemene definitie van autonomie.

In dit onderzoek definiëren we autonoom zijn als *het zelf beheersen van externe invloeden op het eigen besluitvormingsproces.* Hiermee wordt bedoeld dat een agent zelf bepaalt in hoeverre zijn keuzes worden beïnvloed door de omgeving en door andere agenten. In de tegenovergestelde situatie, waarin een agent geen controle heeft over hoe hij zich laat beïvloeden, zullen zijn keuzes volledig afhankelijk zijn van de omgeving en zal de agent door anderen gemanipuleerd kunnen worden. Wij gaan ervan uit dat het beheersen van invloeden een adaptief proces is dat de agent zelf onder controle heeft. Daardoor krijgt autonomie

een verstelbaar karakter. Verstelbare autonomie in onze context betekent *het dynamisch omgaan met externe invloeden op het besluitvormingproces op basis van interne motivaties*. Dat wil zeggen dat de agent kan kiezen om zich open te stellen voor bepaalde invloeden. Doordat agenten de mate van openheid naar elkaar kunnen aanpassen, kunnen ze onderlinge coördinatie bereiken door zich te laten beïnvloeden.

Deze visie op autonomie en verstelbare autonomie stelt bepaalde eisen aan de interne structuur van een agent. In Hoofdstuk 3 van dit proefschrift presenteren we een abstract redeneermodel dat de autonomie van een agent ondersteunt en we geven aan hoe het geïmplementeerd kan worden. De twee belangrijkste componenten van het redeneermodel zijn *het beheersen van invloeden* en *besluitvorming*. Bij de implementatie hebben we voor het besluitvormingsproces van de agenten gebruik gemaakt van besluitvorming op basis van het *Belief-Desire-Intention* (BDI-) model. Een BDI-agent heeft een interne toestand met daarin kennis over de wereld, doelstellingen die hij wil bereiken en mogelijke plannen. Op basis van zijn interne toestand maakt hij keuzes over gewenste acties.

In het proces om beïnvloeding te beheersen bepaalt de agent in hoeverre hij zijn interne toestand laat beïnvloeden door factoren van buitenaf. Hier verwerkt hij nieuwe observaties, informatieberichten en verzoeken op basis van overwegingen als: is dit relevante informatie? Moet ik dit verzoek accepteren? Moet ik dit bericht geloven? Wij stellen voor om redeneerregels te gebruiken om deze prikkels te verwerken. De redeneerregels geven aan welke effecten de prikkels hebben op zijn kennis en doelstellingen in gegeven condities. De kennis en de doelstellingen worden vervolgens weer gebruikt om beslissingen te nemen.

Met dit tweedelig redeneermodel doen we vervolgens een coördinatie-experiment. In een gesimuleerde omgeving moet een brandweerorganisatie een aantal branden blussen. Het doel van de organisatie kan worden bereikt door de brandweermannen aan te sturen welke brand zij moeten blussen. Het aansturen kan op verschillende manieren, variërend van *impliciete*, *emergente coördinatie* tot *expliciete*, *gecentraliseerde coördinatie*. Het experiment laat zien dat onze benadering van autonomie (het beheersen van invloeden) gebruikt kan worden om verschillende vormen van coördinatie te verkrijgen. Bovendien lijkt het goede mogelijkheden te bieden voor dynamische coördinatie. Vragen die opkomen wanneer we individuele agenten controle te geven over hoe ze beïnvloed worden, zijn ondermeer: wat zijn goede vuistregels om externe prikkels te verwerken? Hoe bereiken we dat de agent zijn houding ten opzichte van anderen en van zijn omgeving aanpast op basis van de situatie?

Deze aspecten kunnen we aanpakken door gebruik te maken van de *modulariteit* in het redeneermodel. Ons redeneermodel maakt onderscheid tussen het verwerken van prikkels en het besluitvormingsproces. Doordat we deze processen los van elkaar benaderen, kunnen we domeinonafhankelijke vuistregels ontwikkelen om de prikkels te verwerken. In dit proefschrift noemen we er drie: relevantie van informatie, organisationele kennis en vertrouwen. De eerste twee worden verder uitgewerkt.

In Hoofdstuk 4 behandelen we de mogelijke voordelen van het gebruik van relevantie om invloeden te beheersen. Het idee erachter is dat niet alle informatie belangrijk is voor alle taken. Wanneer je bijvoorbeeld ergens met de auto van A naar B rijdt, is verkeersinformatie over de andere kant van het land totaal niet relevant om je doel te bereiken. Door relevantie van informatie te gebruiken om beïnvloeding te beheersen, kan een agent zich focussen op een bepaalde taak en kan hij zichzelf beschermen tegen overbelasting door teveel informatie. In dit hoofdstuk ontwikkelen we een algoritme om relevantie te bepalen voor BDI-agenten dat gebaseerd is op de *magic sets*-methode die ontwikkeld is om deductieve databases op een efficiënte manier te doorzoeken. We hebben de methode zo aangepast dat deze is toe te passen op het redeneerproces in BDI-agenten.

Het onderwerp van Hoofdstuk 5 is het gebruik van organisationele kennis om externe prikkels te verwerken. Als je deelneemt aan een organisatie, stem je in met bepaalde gedragsregels. Voorbeelden hiervan zijn dat je niet door rood mag rijden, of dat je je baas moet informeren als je laat bent. In dit hoofdstuk laten we zien hoe organisationele gedragsregels vertaald kunnen worden naar redeneerregels om externe prikkels te verwerken. Nieuwe observaties kunnen leiden tot verplichtingen of verboden. Verder laten we zien dat onze aanpak adaptiviteit in de organisatie mogelijk maakt. De regels om prikkels te verwerken kunnen continu gewijzigd worden en daarmee kunnen veranderingen in gedragsregels overgenomen worden door de individuele agenten.

In Hoofdstuk 6 ligt de focus op verstelbare autonomie. We breiden het redeneermodel uit zodat de agenten kunnen redeneren over de vuistregels om invloeden te beheersen. Daarbij moet gedacht worden aan overwegingen als: Zijn de organisationele gedragsregels wel waardevol? Hoe om te gaan met tegenstrijdige gedragsregels? Vraagt een bepaald doel extra focus? De modulariteit in ons redeneermodel garandeert dat de regels om prikkels te verwerken, expliciet gedefinieerd worden. Dit maakt het mogelijk dat er over geredeneerd kan worden. We presenteren een meta-redeneermodel waarmee een agent de gewenste houding ten opzichte van zijn omgeving kan selecteren en kan aannemen. Zo kan hij bijvoorbeeld kiezen om op een bepaalde taak te focussen als de deadline nadert, of hij kan kiezen om hoe dan ook de organisationele regels op te volgen. Met dit proces krijgt de agent de beheersing over hoe hij wordt beïnvloed. Daarmee voldoet hij aan de eisen van autonomie en verstelbare autonomie.

In een simulatie-experiment implementeren we wederom een brandweerorganisatie en we laten zien dat de organisatie dynamische coördinatie vertoont dankzij de zelf-verstelbare autonomie van de brandweermannen. Met dit experiment tonen we aan dat individuen voor een belangrijk deel bijdragen aan het aanpassingsvermogen van een organisatie door zelf te bepalen hoe ze beïnvloed worden. De kunstmatige actoren in ons experiment kunnen omgaan met verschillende vormen van samenwerking en kunnen initiatief nemen om de vorm van coördinatie aan te passen. Dit creëert mogelijkheden om systemen te ontwikkelen waarin mensen en kunstmatige actoren of kunstmatige actoren onderling op een dynamische manier met elkaar samenwerken.

Dankwoord

Het is grappig om onderzoek te doen naar de relatie tussen autonomie en coördinatie. Gedurende het hele proces merk je uit de praktijk dat je tussen de twee moet balanceren. Op bijeenkomsten gebeurde het meerdere keren dat we opmerkten dat er daadwerkelijk sprake was van verstelbare autonomie. Zo moet je om gezamelijk een paper te schrijven afpsraken met elkaar maken. Maar als vervolgens je begeleiders erg druk zijn, kan het handig zijn om zelf initiatief te nemen en zelfstandiger te werken. Of als je verstrikt raakt in technische details, kan een discussie met begeleiders je op een positieve manier beïnlvoeden. Als er deadlines overschreden werden, hadden we al snel een uitleg voor handen.

We zijn praktijkvoorbeelden tegengekomen van verschillende theoretische modellen die in dit proefschrift gepresenteerd worden: het inschatten wat nou echt relevante informatie is, het overwegen of je je voor één keer aan vaste regels mag onttrekken, en het belang van vertrouwen. Het werk aan dit proefschrift heeft mij veel geleerd over de toegevoegde waarde van onderling begrip en flexibiliteit.

Om te beginnen wil ik mij co-promotor, Frank Dignum, bedanken. Veel van de ideeën die hier gepresenteerd worden, zijn voortgekomen uit onze discussies. Ik heb onze samenwerking erg gewaardeerd en ik heb veel van jou geleerd. Wie weet, kunnen we in de toekomst onze samenwerking elders voortzetten.

Vervolgens wil ik mijn promotor bedanken, John-Jules Meyer. Jij bent altijd positief over mijn werk is geweest, ook bij tegenslagen. Ik heb vaak een goed gevoel over gehouden aan onze gesprekken. Je hebt altijd positief meegekeken, zowel naar het overzicht van mijn onderzoek, als naar de tekstuele details.

Bij TNO heeft Martijn Neef een belangrijke rol gespeeld. Jouw praktische blik en jouw heldere manier van uitleggen hielp mij er altijd aan herinneren waarom ik het theoretische werk ook al weer deed. Het bevalt prima om met jou een kamer met je te delen, en ook qua koffiefrequentie voelen we elkaar prima aan.

Thanks to the members of my reading comittee: Jeff Bradshaw, Cristiano Castelfranchi, Catholijn Jonker, Peter Werkhoven and Bob Wielinga. It was an honor for me to have you in my committee and to receive your approval. And thank you for the useful comments.

Het grootste gedeelte van dit onderzoek is gedaan bij TNO in Den Haag. Bedankt André, mijn interne begeleider in de eerste twee jaar. We zijn elkaar sindsdien een beetje uit het oog verloren, maar als ik een patent nodig heb, weet

ik je te vinden. Verder heb ik een prettige tijd gehad met alle collega's van de C2&IV groep en daar om heen, en ik voorzie een productieve en plezierige samenwerking de komende tijd.

Mijn collega's van de IS groep aan de Universiteit Utrecht; de groep van aio's en staf die ik voorbij heb zien komen is te groot om iedereen individueel te noemen. Dankzij jullie heb ik met plezier in Utrecht gewerkt. Zo waren daar de lunch-, koffie- en tafeltennispauzes, en niet te vergeten de conferentiereizen. Ik verwacht een aantal van jullie in een andere omgeving tegen te blijven komen. Virginia, jouw proefschrift is een inspiratie voor mijn werk geweest, en ik wil je bedanken voor het meeschrijven aan een gezamelijk paper.

Verder heb ik het leuk gehad bij bijeenkomsten van SIKS onderzoeksschool en van het ICIS project. Van iedereen daar wil Mattijs graag noemen, collega aio van de UvA; we hebben meerdere vrijdagmiddagen gespendeerd met besprekingen over werk en de tour. Ik heb het zeer gewaardeerd!

Voor dat ik verder ga, wil ik iedereen bedanken die op wat voor manier dan ook interesse heeft getoond in mijn werk.

Werk is één, maar je kunt niet presteren zonder afleiding. Sport! Alle ultimate frisbeespelers van Ultimus Prime en Force Elektro met wie ik afleiding en plezier gevonden heb bij trainigen, wedstrijden en toernooien: bedankt daarvoor. En vrienden! Ik mag de 6 Noord Ultras niet overslaan vanwege alle frituuravondjes met bier en voetbal, en de vakantietripjes. Alle vrienden van de KI-studie; het is altijd leuk om af te spreken voor die good-old Groningen-sfeer. En veel dank aan alle anderen die niet in deze hokjes passen, maar die de afgelopen vier jaar voor prettig gezelschap hebben gezorgd.

En uiteraard dank aan mijn familie, verspreid over het hele land. Jeroen, Caroline, jullie wonen het dichtst bij en jullie vertegenwoordigen hier mijn ooms en tantes. Neven en nichten, ik vind het altijd leuk om met jullie af te spreken. En zo ook de 'schoonfamilie': Nanny en Ernie, geven jullie het door?

Vader, jij hebt mij altijd gestimuleerd, je bent een voorbeeld voor mij. Ward, Reinild en Louise, ik ben trots dat ik jullie mijn broer en zussen mag noemen. Evelien, jij bent de liefste, jij bent onvervangbaar. Ik heb jullie vertrouwen altijd gevoeld, dank voor alle steun!

SIKS Dissertation Series

1998

Johan van den Akker, *DEGAS - An Active, Temporal Database of Autonomous Objects*, CWI, 1998-1

Floris Wiesman, *Information Retrieval by Graphically Browsing Meta-Information*, UM, 1998-2

Ans Steuten, *A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective*, TUD, 1998-3

Dennis Breuker, *Memory versus Search in Games*, UM, 1998-4

E.W. Oskamp, *Computerondersteuning bij Straftoemeting*, RUL, 1998-5

1999

Mark Sloof, *Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products*, VU, 1999-1

Rob Potharst, *Classification using decision trees and neural nets*, EUR, 1999-2

Don Beal, *The Nature of Minimax Search*, UM, 1999-3

Jacques Penders, *The practical Art of Moving Physical Objects*, UM, 1999-4

Aldo de Moor, *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*, KUB, 1999-5

Niek J.E. Wijngaards, *Re-design of compositional systems*, VU, 1999-6

David Spelt, *Verification support for object database design*, UT, 1999-7

Jacques H.J. Lenting, *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.*, UM, 1999-8

2000

Frank Niessink, *Perspectives on Improving Software Maintenance*, VU, 2000-1

Koen Holtman, *Prototyping of CMS Storage Management*, TUE, 2000-2

Carolien M.T. Metselaar, *Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief*, UVA, 2000-3

Geert de Haan, *ETAG, A Formal Model of Competence Knowledge for User Interface Design*, VU, 2000-4

Ruud van der Pol, *Knowledge-based Query Formulation in Information Retrieval*, UM, 2000-5

Rogier van Eijk, *Programming Languages for Agent Communication*, UU, 2000-6

Niels Peek, *Decision-theoretic Planning of Clinical Patient Management*, UU, 2000-7

Veerle Coup, *Sensitivity Analyis of Decision-Theoretic Networks*, EUR, 2000-8

Florian Waas, *Principles of Probabilistic Query Optimization*, CWI, 2000-9

Niels Nes, *Image Database Management System Design Considerations, Algorithms and Architecture*, CWI, 2000-10

Jonas Karlsson, *Scalable Distributed Data Structures for Database Management*, CWI, 2000-11

2001

Silja Renooij, *Qualitative Approaches to Quantifying Probabilistic Networks*, UU,

2001-1

Koen Hindriks, *Agent Programming Languages: Programming with Mental Models*, UU, 2001-2

Maarten van Someren, *Learning as problem solving*, UvA, 2001-3

Evgueni Smirnov, *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*, UM, 2001-4

Jacco van Ossenbruggen, *Processing Structured Hypermedia: A Matter of Style*, VU, 2001-5

Martijn van Welie, *Task-based User Interface Design*, VU, 2001-6

Bastiaan Schonhage, *Diva: Architectural Perspectives on Information Visualization*, VU, 2001-7

Pascal van Eck, *A Compositional Semantic Structure for Multi-Agent Systems Dynamics*, VU, 2001-8

Pieter Jan 't Hoen, *Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes*, RUL, 2001-9

Maarten Sierhuis, *Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design*, UvA, 2001-10

Tom M. van Engers, *Knowledge Management: The Role of Mental Models in Business Systems Design*, VUA, 2001-11

2002

Nico Lassing, *Architecture-Level Modifiability Analysis*, VU, 2002-01

Roelof van Zwol, *Modelling and searching web-based document collections*, UT, 2002-02

Henk Ernst Blok, *Database Optimization Aspects for Information Retrieval*, UT, 2002-03

Juan Roberto Castelo Valdueza, *The Discrete Acyclic Digraph Markov Model in Data Mining*, UU, 2002-04

Radu Serban, *The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents*, VU, 2002-05

Laurens Mommers, *Applied legal epistemology; Building a knowledge-based ontology of the legal domain*, UL, 2002-06

Peter Boncz, *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*, CWI, 2002-07

Jaap Gordijn, *Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas*, VU, 2002-08

Willem-Jan van den Heuvel, *Integrating Modern Business Applications with Objectified Legacy Systems*, KUB, 2002-09

Brian Sheppard, *Towards Perfect Play of Scrabble*, UM, 2002-10

Wouter C.A. Wijngaards, *Agent Based Modelling of Dynamics: Biological and Organisational Applications*, VU, 2002-11

Albrecht Schmidt, *Processing XML in Database Systems*, UVA, 2002-12

Hongjing Wu, *A Reference Architecture for Adaptive Hypermedia Applications*, TUE, 2002-13

Wieke de Vries, *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*, UU, 2002-14

Rik Eshuis, *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*, UT, 2002-15

Pieter van Langen, *The Anatomy of Design: Foundations, Models and Applications*, VU, 2002-16

Stefan Manegold, *Understanding, Modeling, and Improving Main-Memory Database Performance*, UVA, 2002-17

2003

Heiner Stuckenschmidt, *Onotology-Based Information Sharing In Weakly Structured Environments*, VU, 2003-1

Jan Broersen, *Modal Action Logics for Reasoning About Reactive Systems*, VU, 2003-02

Martijn Schuemie, *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*, TUD, 2003-03

Milan Petkovic, *Content-Based Video Retrieval Supported by Database Technology*, UT, 2003-04

Jos Lehmann, *Causation in Artificial Intelligence and Law - A modelling approach*, UVA, 2003-05

Boris van Schooten, *Development and specification of virtual environments*, UT, 2003-06

Machiel Jansen, *Formal Explorations of Knowledge Intensive Tasks*, UvA, 2003-07

Yongping Ran, *Repair Based Scheduling*, UM, 2003-08

Rens Kortmann, *The resolution of visually guided behaviour*, UM, 2003-09

Andreas Lincke, *Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture*, UvT, 2003-10

Simon Keizer, *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*, UT, 2003-11

Roeland Ordelman, *Dutch speech recognition in multimedia information retrieval*, UT, 2003-12

Jeroen Donkers, *Nosce Hostem - Searching with Opponent Models*, UM, 2003-13

Stijn Hoppenbrouwers, *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*, KUN, 2003-14

Mathijs de Weerdt, *Plan Merging in Multi-Agent Systems*, TUD, 2003-15

Menzo Windhouwer, *Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses*, CWI, 2003-16

David Jansen, *Extensions of Statecharts with Probability, Time, and Stochastic Timing*, UT, 2003-17

Levente Kocsis, *Learning Search Decisions*, UM, 2003-18

2004

Virginia Dignum, *A Model for Organizational Interaction: Based on Agents, Founded in Logic*, UU, 2004-01

Lai Xu, *Monitoring Multi-party Contracts for E-business*, UvT, 2004-02

Perry Groot, *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*, VU, 2004-03

Chris van Aart, *Organizational Principles for Multi-Agent Architectures*, UVA, 2004-04

Viara Popova, *Knowledge discovery and monotonicity*, EUR, 2004-05

Bart-Jan Hommes, *The Evaluation of Business Process Modeling Techniques*, TUD, 2004-06

Elise Boltjes, *Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes*, UM, 2004-07

Joop Verbeek, *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise*, UM, 2004-08

Martin Caminada, *For the Sake of the Argument; explorations into argument-based reasoning*, VU, 2004-09

Suzanne Kabel, *Knowledge-rich indexing of learning-objects*, UVA, 2004-10

Michel Klein, *Change Management for Distributed Ontologies*, VU, 2004-11

The Duy Bui, *Creating emotions and facial expressions for embodied agents*, UT, 2004-12

Wojciech Jamroga, *Using Multiple Models of Reality: On Agents who Know how to Play*, UT, 2004-13

Paul Harrenstein, *Logic in Conflict. Logical Explorations in Strategic Equilibrium*, UU, 2004-14

Arno Knobbe, *Multi-Relational Data Mining*, UU, 2004-15

Federico Divina, *Hybrid Genetic Relational Search for Inductive Learning*, VU, 2004-16

Mark Winands, *Informed Search in Complex Games*, UM, 2004-17

Vania Bessa Machado, *Supporting the Construction of Qualitative Knowledge Models*, UvA, 2004-18

Thijs Westerveld, *Using generative probabilistic models for multimedia retrieval*, UT, 2004-19

Madelon Evers, *Learning from Design: facilitating multidisciplinary design teams*, Nyenrode, 2004-20

2005

Floor Verdenius, *Methodological Aspects of Designing Induction-Based Applications*, UVA, 2005-01

Erik van der Werf, *AI techniques for the game of Go*, UM, 2005-02

Franc Grootjen, *A Pragmatic Approach to the Conceptualisation of Language*, RUN, 2005-03

Nirvana Meratnia, *Towards Database Support for Moving Object data*, UT, 2005-04

Gabriel Infante-Lopez, *Two-Level Probabilistic Grammars for Natural Language Parsing*, UVA, 2005-05

Pieter Spronck, *Adaptive Game AI*, UM, 2005-06

Flavius Frasincar, *Hypermedia Presentation Generation for Semantic Web Information Systems*, TUE, 2005-07

Richard Vdovjak, *A Model-driven Approach for Building Distributed Ontology-based Web Applications*, TUE, 2005-08

Jeen Broekstra, *Storage, Querying and Inferencing for Semantic Web Languages*, VU, 2005-09

Anders Bouwer, *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*, UVA, 2005-10

Elth Ogston, *Agent Based Matchmaking and Clustering - A Decentralized Approach to Search*, VU, 2005-11

Csaba Boer, *Distributed Simulation in Industry*, EUR, 2005-12

Fred Hamburg, *Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen*, UL, 2005-13

Borys Omelayenko, *Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics*, VU, 2005-14

Tibor Bosse, *Analysis of the Dynamics of Cognitive Processes*, VU, 2005-15

Joris Graaumans, *Usability of XML Query Languages*, UU, 2005-16

Boris Shishkov, *Software Specification Based on Re-usable Business Components*, TUD, 2005-17

Danielle Sent, *Test-selection strategies for probabilistic networks*, UU, 2005-18

Michel van Dartel, *Situated Representation*, UM, 2005-19

Cristina Coteanu, *Cyber Consumer Law, State of the Art and Perspectives*, UL, 2005-20

Wijnand Derks, *Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics*, UT, 2005-21

2006

Samuil Angelov, *Foundations of B2B Electronic Contracting*, TUE, 2006-01

Cristina Chisalita, *Contextual issues in the design and use of information technology in organizations*, VU, 2006-02

Noor Christoph, *The role of metacognitive skills in learning to solve problems*, UVA, 2006-03

Marta Sabou, *Building Web Service Ontologies*, VU, 2006-04

Cees Pierik, *Validation Techniques for Object-Oriented Proof Outlines*, UU, 2006-05

Ziv Baida, *Software-aided Service Bundling – Intelligent Methods & Tools for Graphical Service Modeling*, VU, 2006-06

Marko Smiljanic, *XML schema matching – balancing efficiency and effectiveness by means of clustering*, UT, 2006-07

Eelco Herder, *Forward, Back and Home Again – Analyzing User Behavior on the Web*, UT, 2006-08

Mohamed Wahdan, *Automatic Formulation of the Auditor's Opinion*, UM, 2006-09

Ronny Siebes, *Semantic Routing in Peer-to-Peer Systems*, VU, 2006-10

Joeri van Ruth, *Flattening Queries over Nested Data Types*, UT, 2006-11

Bert Bongers, *Interactivation – Towards an e-cology of people, our technological environment, and the arts*, VU, 2006-12

Henk-Jan Lebbink, *Dialogue and Decision Games for Information Exchanging Agents*,

UU, 2006-13

Johan Hoorn, *Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change*, VU, 2006-14

Rainer Malik, *CONAN: Text Mining in the Biomedical Domain*, UU, 2006-15

Carsten Riggelsen, *Approximation Methods for Efficient Learning of Bayesian Networks*, UU, 2006-16

Stacey Nagata, *User Assistance for Multitasking with Interruptions on a Mobile Device*, UU, 2006-17

Valentin Zhizhkun, *Graph transformation for Natural Language Processing*, UVA, 2006-18

Birna van Riemsdijk, *Cognitive Agent Programming: A Semantic Approach*, UU, 2006-19

Marina Velikova, *Monotone models for prediction in data mining*, UvT, 2006-20

Bas van Gils, *Aptness on the Web*, RUN, 2006-21

Paul de Vrieze, *Fundaments of Adaptive Personalisation*, RUN, 2006-22

Ion Juvina, *Development of Cognitive Model for Navigating on the Web*, UU, 2006-23

Laura Hollink, *Semantic Annotation for Retrieval of Visual Resources*, VU, 2006-24

Madalina Drugan, *Conditional log-likelihood MDL and Evolutionary MCMC*, UU, 2006-25

Vojkan Mihajlovic, *Score Region Algebra: A Flexible Framework for Structured Information Retrieval*, UT, 2006-26

Stefano Bocconi, *Vox Populi: generating video documentaries from semantically annotated media repositories*, CWI, 2006-27

Borkur Sigurbjornsson, *Focused Information Access using XML Element Retrieval*, UVA, 2006-28

2007

Kees Leune, *Access Control and Service-Oriented Architectures*, UvT, 2007-01

Wouter Teepe, *Reconciling Information Exchange and Confidentiality: A Formal Approach*, RUG, 2007-02

Peter Mika, *Social Networks and the Semantic Web*, VU, 2007-03

Jurriaan van Diggelen, *Achieving Semantic Interoperability in Multi-agent Systems: A Dialogue-based Approach*, UU, 2007-04

Bart Schermer, *Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance*, UL, 2007-05

Gilad Mishne, *Applied Text Analytics for Blogs*, UVA, 2007-06

Natasa Jovanoviç, *To Who It May Concern - Addressee Identification in Face-to-Face Meetings*, UT, 2007-08

Mark Hoogendoom, *Modeling of Change in Multi-Agent Organizations*, VU, 2007-09

David Mobach, *Agent-Based Mediated Service Negotiation*, VU, 2007-09

Huib Aldewereld, *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*, UU, 2007-10

Natalia Stash, *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*, TUE, 2007-11

Marcel van Gerven, *Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty*, RUN, 2007-12

Rutger Rienks, *Meetings in Smart Environments; Implications of Progressing Technology*, UT, 2007-13

Niek Bergboer, *Context-Based Image Analysis*, UM, 2007-14

Joyca Lacroix, *NIM: a Situated Computational Memory Model*, UM, 2007-15

Davide Grossi, *Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems*, UU, 2007-16

Theodore Charitos, *Reasoning with Dynamic Networks in Practice*, UU, 2007-17

Bart Orriens, *On the development an management of adaptive business collaborations*, UvT, 2007-18

David Levy, *Intimate relationships with artificial partners*, UM, 2007-19

Slinger Jansen, *Customer Configuration Updating in a Software Supply Network*, UU, 2007-20

Karianne Vermaas, *Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005*, UU, 2007-21

Zlatko Zlatev, *Goal-oriented design of value and process models from patterns*, UT, 2007-22

Peter Barna, *Specification of Application Logic in Web Information Systems*, TUE, 2007-23

Georgina Ramrez Camps, *Structural Features in XML Retrieval*, CWI, 2007-24

Joost Schalken, *Empirical Investigations in Software Process Improvement*, VU, 2007-25

2008

Katalin Boer-Sorbn, *Agent-Based Simulation of Financial Markets: A modular,continuous-time approach*, EUR, 2008-01

Alexei Sharpanskykh, *On Computer-Aided Methods for Modeling and Analysis of Organizations*, VU, 2008-02

Vera Hollink, *Optimizing hierarchical menus: a usage-based approach*, UVA, 2008-03

Ander de Keijzer, *Management of Uncertain Data - towards unattended integration*, UT, 2008-04

Bela Mutschler, *Modeling and simulating causal dependencies on process-aware information systems from a cost perspective*, UT, 2008-05

Arjen Hommersom, *On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective*, RUN, 2008-06

Peter van Rosmalen, *Supporting the tutor in the design and support of adaptive e-learning*, OU, 2008-07

Janneke Bolt, *Bayesian Networks: Aspects of Approximate Inference*, UU, 2008-08

Christof van Nimwegen, *The paradox of the guided user: assistance can be counter-effective*, UU, 2008-09

Wauter Bosma, *Discourse oriented summarization*, UT, 2008-10

Vera Kartseva, *Designing Controls for Network Organizations: A Value-Based Approach*, VU, 2008-11

Jozsef Farkas, *A Semiotically Oriented Cognitive Model of Knowledge Representation*, RUN, 2008-12

Caterina Carraciolo, *Topic Driven Access to Scientific Handbooks*, UVA, 2008-13

Arthur van Bunningen, *Context-Aware Querying; Better Answers with Less Effort*, UT, 2008-14

Martijn van Otterlo, *The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains*, UT, 2008-15

Henriette van Vugt, *Embodied agents from a user's perspective*, VU, 2008-16

Martin Op 't Land, *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*, TUD, 2008-17

Guido de Croon, *Adaptive Active Vision*, UM, 2008-18

Henning Rode, *From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search*, UT, 2008-19

Rex Arendsen, *Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven*, UVA, 2008-20

Krisztian Balog, *People Search in the Enterprise*, UVA, 2008-21

Henk Koning, *Communication of IT-Architecture*, UU, 2008-22

Stefan Visscher, *Bayesian network models for the management of ventilator-associated pneumonia*, UU, 2008-23

Zharko Aleksovski, *Using background knowledge in ontology matching*, VU, 2008-24

Geert Jonker, *Efficient and Equitable Exchange in Air Traffic Management Plan*

Repair using Spender-signed Currency, UU, 2008-25

Marijn Huijbregts, *Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled*, UT, 2008-26

Hubert Vogten, *Design and Implementation Strategies for IMS Learning Design*, OU, 2008-27

Ildiko Flesch, *On the Use of Independence Relations in Bayesian Networks*, RUN, 2008-28

Dennis Reidsma, *Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans*, UT, 2008-29

Wouter van Atteveldt, *Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content*, VU, 2008-30

Loes Braun, *Pro-Active Medical Information Retrieval*, UM, 2008-31

Trung H. Bui, *Toward Affective Dialogue Management using Partially Observable Markov Decision Processes*, UT, 2008-32

Frank Terpstra, *Scientific Workflow Design; theoretical and practical issues*, UVA, 2008-33

Jeroen de Knijf, *Studies in Frequent Tree Mining*, UU, 2008-34

Ben Torben Nielsen, *Dendritic morphologies: function shapes structure*, UvT, 2008-35

2009

Rasa Jurgelenaite, *Symmetric Causal Independence Models*, RUN, 2009-01

Willem Robert van Hage, *Evaluating Ontology-Alignment Techniques*, VU, 2009-02

Hans Stol, *A Framework for Evidence-based Policy Making Using IT*, UvT, 2009-03

Josephine Nabukenya, *Improving the Quality of Organisational Policy Making using Collaboration Engineering*, RUN, 2009-04

Sietse Overbeek, *Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality*, RUN, 2009-05

Muhammad Subianto, *Understanding Classification*, UU, 2009-06

Ronald Poppe, *Discriminative Vision-Based Recovery and Recognition of Human Motion*, UT, 2009-07

Volker Nannen, *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*, VU, 2009-08

Benjamin Kanagwa, *Design, Discovery and Construction of Service-oriented Systems*, RUN, 2009-09

Jan Wielemaker, *Logic programming for knowledge-intensive interactive applications*, UVA, 2009-10

Alexander Boer, *Legal Theory, Sources of Law & the Semantic Web*, UVA, 2009-11

Peter Massuthe, *Perating Guidelines for Services*, TUE, Humboldt-Universitaet zu Berlin, 2009-12

Steven de Jong, *Fairness in Multi-Agent Systems*, UM, 2009-13

Maksym Korotkiy, *From ontology-enabled services to service-enabled ontologies*, VU, 2009-14 making ontologies work in e-science with ONTO-SOA

Rinke Hoekstra, *Ontology Representation - Design Patterns and Ontologies that Make Sense*, UVA, 2009-15

Fritz Reul, *New Architectures in Computer Chess*, UvT, 2009-16

Laurens van der Maaten, *Feature Extraction from Visual Data*, UvT, 2009-17

Fabian Groffen, *Armada, An Evolving Database System*, CWI, 2009-18

Valentin Robu, *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets*, CWI, 2009-19

Bob van der Vecht, *Adjustable Autonomy: Controling Influences on Decision Making*, UU, 2009-20