Technologie-Zentrum Informatik

Technical Report 35

Strategic Management of Autonomous Software Systems

– Overview Article –

Ingo Johannes Timm

TZI-Bericht Nr. 35
2006

Universität Bremen

TZI-Berichte

Herausgeber:
Technologie-Zentrum Informatik
Universität Bremen
Am Fallturm 1
28359 Bremen
Telefon: +49-421-218-7272
Fax: +49-421-218-7820
E-Mail: info@tzi.de
http://www.tzi.de

ISSN 1613-3773

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software engineering of systems and processes is continuously increasing in complexity of design and functionality. One possibility to cope with the resulting requirements is the application of distributed software systems. The rapid establishment of the Internet and networking technologies sped up the significance of distributed computing, which is in need of high-sophisticated engineering methodologies. In Artificial Intelligence (AI), distributed problem-solving is considered as a possibility for scaling up AI-techniques. These trends in computer science are accompanied by similar developments in business applications. Global competition results in the demand for shorter product life cycles and forces companies to cut down development and delivery times. In this context, innovative information technologies in general and flexible as well as adaptive concepts are of increasing importance for business success. The requirements for dynamic decision making in this context result in a mutual contradiction: Increasing autonomy creates optimal structures and processes on a micro level, but often fails to optimize the system as a whole, which is the goal of strategic consideration on the macro level. Strategic optimization on the macro level on the other hand tends to introduce boundaries for microscopic decisions such that optimal structures cannot evolve or time and support consumptions increases.

This contradiction arises a key challenge in systems engineering. Conventional engineering of monolithic software systems provides clear structures and explicit processes which allow for consistent management rules, plans, and control at any step and level of design and implementation as well as runtime behavior. In various application domains, such as manufacturing or health care logistics, distributed entities are existing in the real-world. Thus a natural approach to cope with the inherent complexity and dynamics of these applications is the implementation of distributed systems. In the early days of distributed software engineering, parallel computing, i.e. scaling of computational power was in the center of research. The inherent requirements of those real-world applications as well as further developments in AI, lead to an increasing amount of ”slack” in execution of actions according to predefined plans. The autonomous behavior of such systems extends from reactive systems over deliberative approaches to autonomous systems. In consequence, the problem of contradictions between autonomy and strong regulation becomes increasingly important in the area of

software engineering.

1.1 Balancing Autonomy and Strategic Management

Intelligent agents represent an area of research in Distributed Artificial Intelligence (DAI) for implementing autonomous systems. These agents are software entities, which can act autonomously, communicate with other entities, are goal-oriented (proactive), and use explicit knowledge about their application domain [Knirsch and Timm, 1999]. They are able to collaborate and solve problems in a distributed manner. The German Priority Research Program funded by the Deutsche Forschungsgemeinschaft (DFG) on ”Intelligent Software Agents and Business Applications” has shown, that agents are a beneficial approach to incorporate flexibility in real-world applications as well as to perform an effective information logistics [Kirn et al., 2006]. In economics, this gain of flexibility by autonomous subsystems is considered to be paid off by a loss of control for the strategic management, such that a contradiction arises between strategic control of the system and autonomous system parts [Dembski and Timm, 2005].

Autonomous decision making in system parts should include the ability to recognize a decision situation, identify alternatives, assess them, decide, and implement the decision. Doing so, increased robustness and positive emergence of the system as a whole should evolve. In consequence, the system should be able to cope with dynamics and complexity, distributively and flexibly. Nevertheless, we assume the local autonomy of distributed software entities may cause suboptimal system states. This assumption is supported by game theory, which can be used to implement autonomous decision-making in agents. Let us assume that the agents are acting in an environment which meets the requirements of non-cooperative games [Holler and Illing, 1990]. Here, the Nash equilibrium theory is used for computing dominant strategies, i.e., strategies which are optimal under the assumption that the opponent acts rationally resp. chooses that strategy, which is maximizing his payoff [Kreps, 1994]. There are many environments resp. games, where none or more than one Nash equilibrium exists, such that there is no dominant strategy [Holler and Illing, 1990]. In the latter case with more than one equilibrium, the outcome resp. chosen strategies of the agents are non-deterministic and those strategies are not necessarily maximizing the agents' profit. In consequence, maleficent system behavior may occur. However, even if there is only one Nash equilibrium, it is not ensured, that the local optimality of an agent is also beneficial to the global optimum of the system. For example, the well-known prisoner's dilemma has the dominant strategy to cooperate even if the strategy of deflection followed by both agents would be superior with respect to the system level [Axelrod, 1997].

Several implications to software systems as a whole arise from that assumption. One of the key challenges can be found in the heterogeneous levels of decision making within autonomous processes, i.e., handling contradictory requirements. With a higher level of autonomy, decision making is supposed to become faster and more flexible, but it may also lead to a loss of control for the management of the system. This gap between operational decision-making and strategic management has to be bridged to

ensure reliable decision-making. Thus, achieving a suitable solution involves organizational (e.g., information and knowledge flow) as well as managerial (e.g., handling contradictions) aspects.

Finally, it does not seem appropriate to construct distributed software entities, which are either purely autonomous or strongly regulated. Both aspects, autonomy and regulation, represent the opposed margins of a scale. The challenge for the engineering of distributed autonomous software is to realize systems with a situation-dependent, optimal mixture of autonomy and regulation. This paper discusses research results on the incorporation of strategic management in autonomous software systems as a proposed solution for balancing autonomy and regulation. Doing so, we deal with different levels of autonomy and regulation: strongly regulated subsystems, and operational, tactical, and strategic autonomy.

1.2 Organization of the Paper

Chapter 2 deals with the different levels of autonomy in software systems. The main stages of autonomy can be partitioned into the levels: operational, tactical, and strategic autonomy. In the next chapter (Chapter 3), the incorporation of strategic management into autonomous systems is proposed. Engineering of autonomous systems is in the focus of the chapter 4. Here we discuss the engineering process for autonomous systems with its specific tasks and challenges. Applications for autonomous systems as well as exemplary approaches are introduced in chapter 5. Finally, we conclude with a discussion of our results (Chapter 6).

Chapter 2

Autonomous Software Systems

In software engineering, systems under research and development are of increasing complexity up to the point where they overwhelm the designers. Therefore, more sophisticated approaches to system engineering and algorithm design are required. The research field of Artificial Intelligence (AI) is dedicated to systems, where it is assumed that intelligence is required for efficient solution [Negnevitsky, 2002]. In the early research on AI, the objective of AI was defined as to develop systems, which solve problems that would require intelligence if solved by humans [Boden, 1977]. [Russell and Norvig, 1994] focuses on AI as the development of intelligent entities:

> The field of artificial intelligence, or AI [...] attempts not just to understand but also to build intelligent entities [Russell and Norvig, 1994, p. 1].

Distributed Artificial Intelligence (DAI) extends AI systems by autonomous and cooperative behavior for scalability, multiple problem-solving strategies, and reusability purposes [Müller, 1993]. [Findler, 1991, p.23] defines the scope of DAI as follows:

> Distributed planning and problem solving systems handle tasks that cannot be dealt with effectively and efficiently by one single processor.

Various authors, for example [Moulin and Chaib-Draa, 1996], claim that DAI and especially the multiagent technology have significant advantages over a single, monolithic problem solver with respect to ”faster problem solving by exploiting parallelism; decreased communication by transmitting only high-level partial solutions to other agents rather than raw data to a central site; more flexibility by having agents with different abilities dynamically team up to solve current problems; and increased reliability by allowing agents to take on responsibilities of agents that fail” [Moulin and Chaib-Draa, 1996, p.5]. In the research field of multiagent systems, a key perspective lies on the analysis and engineering of autonomy [Weiss, 1999], [Wooldridge, 2002].

Considerable progress has been made in the interdisciplinary research on autonomy (cf. German Priority Research Program on ”Socionics” funded by the DFG, [Nickles et al., 2002]). [Weiss et al., 2005, p. 1] introduce autonomy as an emerging software property:

In addition to that, the increasing complexity of software in domains like e/m-commerce, telecommunications, logistics, knowledge management, and simulation of social and economic processes on the one hand and the identification of autonomy as an enabler for emerging information processing paradigms such as grid computing, (web-)service-oriented computing or ubiquitous computing on the other have given rise to a more general interest in autonomy as a software property.

2.1 Properties of Autonomy

The definition of autonomy has always been discussed in DAI. There are different approaches to define it. A naive approach to the definition of autonomy uses the external view on a system. A system is identified as autonomous, if it is acting non-deterministically, i.e., the system acts differently in two identical situations. However, this does not mean, that an autonomous system has to be non-deterministic. The appearance of non-determinism arises from the limited view on the environmental state (situation). If the internal state of the system is included, an autonomous system might also be deterministic. A better approach to define autonomy resp. autonomous systems is the consideration of properties. In this context, autonomy resp. autonomous systems is best described by the three properties: pro-activity, interaction, and emergence. These properties are the building blocks for autonomous subsystems as they are described below:

- *pro-activity*
  The decision of a subsystem (actor) is not only performed on the basis of hard-wired input-output schemes. Furthermore the actor is capable of interpreting the environment. Doing so, the system activates goals resp. initiates actions without specific external events. Therefore, the actors require the ability to reason about its goals and the current situation, i.e., an explicit representation of goals and environment is required.

- *interaction*
  The autonomous system is capable of interacting with its environment. This includes the perception of and the interaction with the environment as well as the communication with other autonomous systems. The actor should increase the individual utility as well as indirectly the utility of the overall system. A fundamental assumption for interaction is, that there is some ”slack” in the coordination resp. interaction of autonomous agents.

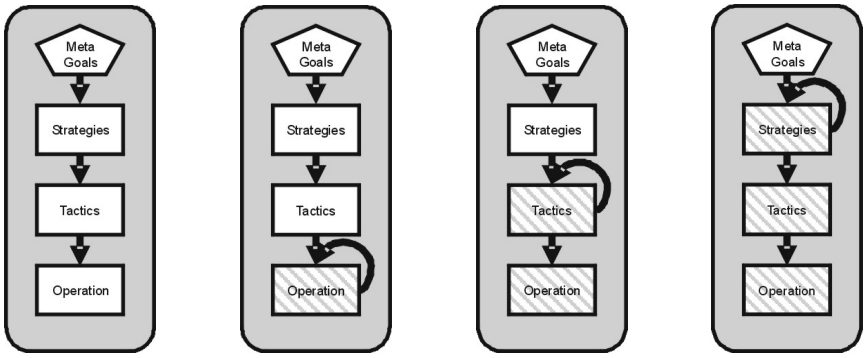
- *emergence*
  The elements introduced so far, pro-activity and interaction, are properties of autonomous systems, which have to be implemented within an actor. If there are more than one autonomous subsystems building a larger system, the larger system should contain properties, which emerge from the interaction of locally resp. pro-actively acting subsystems. The naive formulation of this fundamental assumption is, that the system is more than the sum of this parts.

In our research, the understanding of emergence and its aspects is essential for the consideration of autonomous software systems on a strategic level. In the following, the different perspectives of emergence in DAI are discussed. For further details refer to [Timm et al., 2002] and [Timm et al., 2001d]. Emergent behavior became one of the most important criteria for the evaluation of autonomous software systems like multiagent systems from the very beginning in agent research. Most authors use this term in a metaphorical and undefined manner. Here, a definition based on three major aspects will be used. The first aspect is focused on emergent properties as a large-scale effect of locally interacting autonomous systems: ”Emergent properties are often surprising because it can be hard to anticipate the full consequences of even simple forms of interaction” [Axelrod, 1997, p. 4]. Jacques Ferber’s view is more centralized on emergent organization. ”They [organisational structures] can also be defined a posteriori, and we then speak of emergent organisations. These most frequently contain only reactive agents and are characterised by the absence of any predefined organisational structure, their structure being entirely the result of interactions between agents. In this case, positions and relationships are not determined in advance, but appear as the product of the behaviours of each of the agents. More precisely, the distribution of functions and tasks follows an auto-organisation procedure, which permits an organisation to evolve and to adapt easily to modifications in the environment and to the needs of a group of agents” [Ferber, 1999, p. 114]. The third aspect links emergence with the transition from reactive agents to deliberative ones. Doing so ”the idea that intelligent behavior emerges from the interaction of various simpler behaviors” [Wooldridge, 1999, p. 49] arises within this theoretical basis.

A closer look at all three aspects shows that interaction resp. communication is the main reason for achieving global structures by local interaction, dynamic organization by simple (communication) rules, and intelligent behavior of an autonomous system as a whole. Considering the emergent behavior of a autonomous systems should result from interaction of the autonomous subsystems. On the one hand the cooperation and coordination follows local optimization criteria (goals), and on the other hand it has to take a joint optimization criterion (goals of autonomous systems) into account. This collaboration should lead to a global optimum of the system (emergent effect).

2.2 Levels of Autonomy

In recent research, there are discussions on different levels of autonomy [Rovatsos and Weiss, 2005], [Müller, 1997], [Falcone and Castelfranchi, 2000]. In early multiagent research, [Castelfranchi and Conte, 1992] discuss a very high degree of autonomy, such as the influence of predefined norms, behavior patterns, or procedures is irrelevant, resp. the relevance is very low with respect to the action-selection-process within an agent. Nevertheless, autonomy is a property, which may lead to partially unwanted system states resulting from conflicting or inconsistent goal sets. The dynamic and complex interdependencies of autonomous subsystems can lead to systems, which are organization emerges at runtime. Thus, software engineers of autonomous systems may not consider any possible constellation of subsystems at design time.



(a) strong regulation (b) operational autonomy (c) tactical autonomy (d) strategic autonomy

Figure 2.1: Levels of autonomy in autonomous systems

The detailed discussion of autonomy in this paper follows a systematic approach using the levels of decision making known from economics and system theory: operations, tactics, and strategies (cf. [Hentze et al., 1993], [Küpper, 1997]). Consequently, we introduce four levels of autonomy (LoA) refining the autonomy-regulation scale mentioned in the introduction: strong regulation, operational autonomy, tactical autonomy, strategic autonomy (cf. Figure 2.1). Autonomous systems are situated in an environment with the capability to perceive and interact with it. The complexity of the deliberation process within an agent is strongly related to different environmental properties. In [Russell and Norvig, 1994] and refined in [Russell and Norvig, 2003], the following properties for the classification of agent environments are proposed:

- *observable*
  The property *observable* indicates whether any information of the environment is accessible to any agent (full, partial)

- *deterministic*
  This a property of the environment, which specifies whether an action has a single guaranteed effect (deterministic, stochastic)

- *episodic*
  In an *episodic* environment, the performance of an agent is based on a discrete episode, i.e., there is no link between the performance of the agent which is involved in multiple scenarios. In contrast to episodic environments, sequential environments does not consist of independent scenarios resp. any action of the agent may have consequences in any scenario where the agent is involved in

- *static*
  The dynamics of the environment is described by this property. In static environments, any changes are caused by actions of agents. Semi static environments contain changes within the autonomous systems which are not necessarily

grounded on the perception of the environment. Finally, dynamic environments contain operation processes next to the agents, which potentially change the environment.

- *discrete*
  Further differentiation of environments follows the properties *discrete* or *continuous*. In discrete environments, e.g., games like chess, actions and perception are arranged in some kind of rounds. In continuous environments there is no structure in acting and perceiving, e.g., robot navigation.

- *agents*
  This property distinguishes between environments where only one (single agent) or multiple autonomous systems (multi agent) are involved.

In the following, we use this categorization to specify the levels of autonomy and to describe their respective adequacy in application domains. Table 2.1 summarizes the levels of autonomy with respect to the environment properties. Within this paper, we use the BDI architecture for intelligent agents and the Discourse Agents as an extension as an example for the implementation of different levels of autonomy [Rao and Georgeff, 1995].

LoA	observable	deterministic	episodic	static	agents
0: strong regulation	full	deterministic	episodic	static	single
1: operational autonomy	partial	deterministic	episodic	static	multi
2: tactical autonomy	partial	stochastic	episodic	semi	multi
3: strategic autonomy	partial	stochastic	sequential	dynamic	multi

Table 2.1: Task domains and levels of autonomy

LoA 0: Strong Regulation

Strongly regulated systems are the class of systems usually dealt with in ”traditional” software engineering. In strongly regulated systems, there is no part of the system with autonomous capabilities. Any decision – regardless of the decision level – is predefined or determined by an external entity. Conventional monolithic systems are examples for this class of systems. They proved to be effective in environments with limited complexity characterized by full observability, determinism and episodic structure without further autonomous systems and with a mainly static behavior. Limitations occur, if the autonomous systems are situated in environments where tasks have to be solved, which differ from this setting.

In the manufacturing domain, process planning and production control has been effectively managed and realized by monolithic systems as far as line production with

only few disturbing and unexpected events are in question. However, the trend of individualization reduces the planning horizon in production control, such that even without an increasing number of disturbances monolithic systems reach their structural limitations [Tönshoff et al., 2001a].

LoA 1: Operational Autonomy

The first step of increasing autonomy is associated with the operational level of decision making. Here, an autonomous software system gains the competence to decide on an operational level with a specific ”slack” in the behavior. However, this decision still follows the tactical and strategic boundaries of the system. In agent technology, the operational autonomy may be implemented using reactive agent architectures. In the BDI approach, operational autonomy means that there is no flexibility in the desires or the intention selection mechanism, i.e., the desires and intentions cannot be modified by the agent. However, the agent is capable to reflect or refine plans. In Section 2.4, an approach to operational autonomy will be presented (cf. [Timm, 2004a]) on the basis of adaptive action plans.

With respect to the categories introduced by [Russell and Norvig, 2003], software systems implementing operational autonomy are effective in environments which are partially observable, deterministic, episodic, and static. Obviously, multiagent environments do not prevent operational autonomy. The benefits of operational autonomy are that the underlying algorithms allow for an efficient implementation and immediate response to (episodic) disturbances. However, operational autonomy is restricted to reactions on the basis of short term episodes and therefore does not consider mid- or long-term objectives.

LoA 2: Tactical Autonomy

Tactical autonomy extends operational autonomy with respect to the tactical level. Tactical decision making in autonomous software systems enables the system to deliberate on different alternatives for operational behavior. While in operational autonomy, plans are under consideration, e.g., linearization of partial plans, tactical autonomy is performed at the level of goals resp. intentions. Considering BDI agents, the planning and execution level is associated with the operational autonomy. The deliberation step of a BDI agent, where an agent selects resp. creates an intention with respect to its desires and its current state, is the corresponding level for tactical decision-making. Tactical autonomy includes some kind of algorithmic variety in generating options and intentions on the basis of a current state. In Section 2.5, an approach to enable tactical autonomy on the basis of capability management will be introduced (cf. [Timm and Woelk, 2003a]).

Following the categories of [Russell and Norvig, 2003], software systems incorporating tactical autonomy are able to cope with stochastic and semi-dynamic task domains in addition to the capabilities of operational autonomous systems. The limitations of tactical autonomous systems arise if the environment contains a sequential problem structure, i.e., the dynamics of the environment become part of the task [Russell and Norvig, 2003].

LoA 3: Strategic Autonomy

The strategic aspects represent the highest level of decision making within a software system and are conventionally determined by the system's designer in advance or by external influence, e.g., the user, during runtime. The architecture of each individual agent incorporates static strategies, e.g., by the definition of individual desires and specific algorithms. In conventional BDI, strategic autonomy by the agent itself is not feasible as the desires are determined statically. The selection of desires for the option selection process is based on an accessibility relation, i.e., the agent's beliefs are used for computing a relation identifying those desires, which are accessible by action sequences starting with the current state. In Section 2.6, we introduce an approach for strategic autonomy as an extension to conventional BDI. Here the agents are enabled to dynamically compute interdependencies between desires and intentions with respect to the current state of the agent. The approach is based on conflict management as the interdependencies between desires and intentions are handled as conflicts of interest [Timm, 2004b].

Obviously, strategic autonomy increases the computational complexity within an agent. Additionally strategic capabilities of agents reduce the human control of the entire system. The combination of both aspects results in less acceptance of such systems in special applications for example in medicine. Consequently, it is not useful to apply strategic autonomy for any task in any domain. Following the classification of [Russell and Norvig, 2003], strategic autonomy should be applied in environments where the system is completely dynamic, i.e., where the agent, the environment, and the opponent agents change over time.

2.3 Definition of Discourse Agents

There is a close connection between research on autonomy of software systems and research on multiagent systems, as autonomy is one of the key features attributed to such systems, i.e., software agents are assumed to behave autonomously. The agents act on behalf of other agents or humans, and should therefore act without direct intervention from human users or other entities, i.e., agents should at least have some partial control about their behavior and their internal states [Wooldridge and Jennings, 1995], [Ferber, 1999], [Tecuci, 1998]. Agents are applied to those domains, where complexity and dynamics of the application prevent efficient use of predefined resp. hard-wired behavior. Consequently, agents are designed and implemented with respect to enabling autonomous behavior, i.e., the principal of the agent is not providing an action sequence to the agent but the agent contains planning algorithms to generate action sequences by itself [Kalenka and Jennings, 1997]. Thus, agents are not regulated externally during runtime but contain algorithms and methods for individual planning. Russell & Norvig describe the degree of autonomy in this context as follows [Russell and Norvig, 2003]:

> "To the extent that an agent relies on the prior knowledge of its designer rather than on its own percepts, we say that the agent lacks autonomy." [Russell and Norvig, 2003, p. 37]

The work presented here is based on the formal conceptualization and specification of Discourse Agents introduced in [Timm, 2003] resp. [Timm, 2004b]. The Discourse Agent approach specifies an architecture for agent behavior, knowledge representation, and inferences. This basic approach strictly separates internal and external aspects due to privacy and security issues. Following the MECCA multiagent architecture introduced by [Lux, 1995], the Discourse Agents are based on a three layer architecture consisting of *communicator*, working on a low-level realization of speech acts, *controller*, determining general agent behavior, and *executor*, i.e. an interface to existing components, e.g. enterprise resource planning (ERP) systems and further information sources [Timm, 2004b], [Timm, 2001b].

The communicator should be implemented with respect to standardization efforts, like FIPA [Poslad and Charlton, 2001]. In the case at hand, the communicator is realized on top of a FIPA compliant agent toolkit (JADE, developed by CSELT S.p.A.[1]). The executor is the most domain dependent part of the agent, as it has to be implemented according to its directly connected resources, e.g., machine tools. While the communicator and executor layer is constructed in a straightforward manner, the design of the controller layer is more sophisticated: The controller implements two innovative concepts: conflict-based agent control (cobac) for the strategic autonomy and open, adaptive communication (oac) for the operational autonomy.

The agent deliberation is handled by the controller, i.e., the behavior control of an agent is performed within the controller. The controller collects information by perception and interaction with other agents and can derive new knowledge by inferences. The deliberation process in the controller is based on the deliberative agent architecture BDI [Rao and Georgeff, 1995]. The formal foundation introduces a new (multi-) modal logic, which integrates the formal approaches VSK logic [Wooldridge and Lomuscio, 2000] for inter-agent behavior and the LORA logic [Wooldridge, 2000] for deliberative agent behavior. The Discourse Agent logic is a (multi-)modal sorted first order logic with an underlying branching temporal structure [Timm, 2004b]. For further definitions, let Ω be the set of well-formed formulas with respect to the grammatical definitions of this logic; $\mathcal{B} \subset \Omega$ is denoting the set of possible beliefs. The key concepts for the definition of Discourse Agents are briefly introduced in the following paragraphs. For details please refer to [Timm, 2004b].

1 DEFINITION (DISCOURSEAGENT) *A Discourse Agent is given by a 7-tuple:*
$Ag = \langle L, Act, see, reflect, decide, execute, l_0 \rangle$,
where $l_0 \in \mathcal{L}$ *denotes the initial state.*

While L, Act, and $decide$ are introduced in the next section, see (perception function) , $reflect$ (knowledge revision function), and $execute$ (action selection function) are not in the focus of this paper. As basic concepts within an agent controller we define *actions*, the agent is capable of *plans* as dynamic action sequences and *local states* as explicit state representations.

2 DEFINITION (ACTION) *Let* α^c *be a communicative action executed in the communicator layer and* α^e *an executive action performed in the executive layer, then the sets*

[1] http://sharon.cselt.it/projects/jade/

$Act^c = \{\alpha_0^c, \ldots, \alpha_m^c\}$ and $Act^e = \{\alpha_0^e, \ldots, \alpha_n^e\}$ denote the communicative resp. executive actions of an agent. Together they build the action potential $Act = Act^e \cup Act^c$.

In analogy to AI planning, plans in the Discourse Agent architecture are formally defined as tuples consisting of pre and post conditions, a set of available actions, and the mappings $status$ and $select$.

3 DEFINITION (ACTIONPLAN) *Let $\varphi_{pre}, \varphi_{post} \in \Omega$ be a pre resp. post condition, $\mathcal{B} \subset \Omega$ the set of possible beliefs of the agent, and $A \subset Act$ a set of available actions, then $plan = \langle \varphi_{pre}, \varphi_{post}, A, status, select \rangle$ is called action plan, iff*

- *$status : B \times \varphi_{pre} \times \varphi_{post} \mapsto x$, with $x \in \mathbb{R}_0$ and $B \in \mathcal{B}$, is a mapping denoting the execution status of the plan and*
- *$select : status \times \mathcal{B} \rightarrow A$ is a mapping, which selects an action as the next step of the plan.*

The set Pln denotes the set of all action plans of an agent.

A local state is the core component of BDI agents and is therefore specified using the mental categories *beliefs*, *desires*, and *intentions* as follows:

4 DEFINITION (LOCALSTATE) *The local state is defined as the 5-tuple: $L = \langle B, D, I, Plan, \gamma \rangle$ iff*

- *$B \subset \mathcal{B}$ is the set of current beliefs,*
- *$D \subset \mathcal{B}$ is the set of current desires,*
- *$I \subset D \times Pln$ is the set of active intentions,*
- *$Plan \subset Pln$ is the set of available plans, and*
- *$\gamma : \mathcal{B} \times D \rightarrow \mathbb{R}$ is a mapping computing the relevance of a desire in the current situation*[2].

2.4 Operational Autonomy

As discussed above, operational autonomy enables an autonomous behavior in the execution of plans. In the following, an approach for operational autonomy on the basis of adaptive plans is introduced. For more details see [Timm, 2004a], [Timm, 2004b], [Timm et al., 2001d], and [Timm, 2001a].

In the last section, we introduced action plans on an abstract level. From the BDI perspective, action plans represent the operational level of behavior, i.e., the deliberation process came up with a decision resp. an intention, which is pursued by an action plan. To enable operational behavior, the concept of action plans is specialized. The

[2]The mapping is simplified for this paper, the Discourse Agent approach define a more sophisticated set of mappings, which are assessing desires with respect to user relevance, potential, and risk.

adaptive action plans have been developed to enable flexible communication behavior in agents, i.e., the actions are mapped to message classes in the communicator. However, the concept has been formalized for any action classes including executive action classes. The approach is called open, adaptive communication and extends action plans with learning capabilities. As discussed in Section 2.1, communication is one of the key mechanisms for achieving emergent behavior. Therefore, an adequate selection and configuration of communication protocols is required. Known approaches to agent-oriented analysis and design lack an intuitive methodology to generate and customize agent communication protocols (cf. Section 4.1). Furthermore, communication protocols are often defined within a static structure, which cannot be directly adapted by the agents during runtime. To address this problem we propose an approach of open, adaptive communication protocols (oac). By the oac approach three main extensions of classical concepts are implemented:

- The use of communication is not restricted to a given set of protocols and protocols of opponents do not have to be known (open),
- dialogues do not have static structures only, but are flexible and can be adapted, refined and even synthesized during runtime (adaptive), and
- conflict management determines the strategy for an agent's behavior within a concrete dialogue.

Note that in contrast to agent systems with fixed communication protocols this approach uses a generalized view on dialogues. In classical approaches a dialogue defines an agent communication protocol, e.g., a contract net protocol or a Dutch-auction protocol. The concept of dialogues as action plans integrates communicative as well as executive actions.

The basis of the adaptive plans are Markov models, i.e., temporal homogeneous Markov chains. For the specification of adaptive plans potential actions of the agent are discriminated into a (finite) set of action classes $\overline{\alpha_1}, \overline{\alpha_2}, \overline{\alpha_3}, ..., \overline{\alpha_n}$. The classification of action classes is based on an equivalence relation between different actions. Here, the equivalence relation is simplified by the application of performatives. Two communicative actions are handled as equal if they have the same performative. These action classes form the state space of the Markov processes; the time scale is modeled in a discrete manner $(t = 1, 2, 3, ..)$.

5 DEFINITION (ADAPTIVE PLAN) *Let X_t be the random variable of the process and the $(n \times n)$-matrix with $P = (p_{ij})$ the transition matrix. The corresponding probability distribution is defined by the equation:*

$$Prob(X_t = \overline{\alpha_j} | X_{(t-1)} = \overline{\alpha_i^*}) = p_{ij} \text{ for all } t = 1, 2, 3, ...$$

The current state of the Markov model is indicated by the agent's or other agents' action class $\overline{\alpha_i}$. E.g., actions mean message received or an action of an opponent agent. The correct termination of a plan execution is modeled with a specific action class $\overline{\alpha^f}$. This class contains the terminating actions, i.e., the transition probability from this state

to any other state is 0. The concrete specification of adaptive plan uses two Markov models which form an interaction pattern. One model is used for the representation of incoming events (matrix P^*) and the second model is used as a decision model for the agent (matrix P^*). Each pair of transition matrices specifies a generalized communication protocol. In those protocols, the next action class is not determined directly, but is chosen probabilistically. This approach can also be used for modeling conventional communication protocols (cf. Figure 2.2).

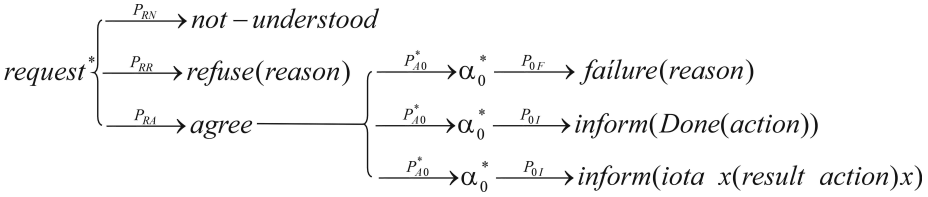


Figure 2.2: FIPA request protocol as Markov chain

The analysis and design of these protocols is done with minimum effort as only required protocol structures must be defined and initial communication protocols must be generated. The agents modify their protocols autonomously during simulation and application as follows:

- **Adaptation**
  The execution of existing communication protocols leads to an adjustment of the selection probabilities (transition probabilities in the model) of the next action due to prior experience with this dialogue partner, the dialogue partner's team, or the overall multiagent system. This adaptation is formalized by multiplication. Note that consequently all zero transition probabilities always keep their value.

- **Refinement**
  By refinement an agent extends protocols by adding new "states" X_i, i.e., speech act types out of a given set of basic communicative acts, e.g., FIPA/ACL. The respective transitions are initialized in a straightforward manner. Another method of refinement is to keep the states as they are, but to implement "new" transitions by setting zero transition probabilities to $P_{ij} > 0$, or to remove certain transitions annulling their probabilities. Refinement is selected if an existing protocol tends not to lead to a satisfying result according to the agent's goals and the multiagent system's goals.

- **Synthesis**
  The automatic generation of protocols is the methodology of choice for the creation of new communication protocols. The basis for this method is a predefined set of dialogue "skeletons" as they occur within known communication protocols. The first step of the synthesis is to select one of them as a core model. The necessary extension and customization of this rudimentary model follows the adaptation and refinement steps described above.

Evaluation

Electronic market-places with local communication-rules are settings where it is assumed that operational autonomy shows a beneficial behavior. Therefore, the evaluation of oac as an approach for operational autonomy is performed in a simulation scenario based on an electronic marketplace with varying communication rules and conventions (conservative, communicative, aggressive). The success criteria of the adaptation on an operational level is measured by the workload of the agent as well as how much it has ”earned” in a virtual currency. In order to gain statistically significant data, simulation runs with comparable settings but also a wide range of settings have been evaluated. This statistical analysis is based on 215 experiments and 1.4 million dialogues.

The hypothesis is that oac is beneficial for realizing operational autonomy for autonomous agents. This means in the context of this scenario, that an agent is capable of adapting to different kind of market places resp. opponent agents. The behavior is compared to agents, which do not implement operational autonomy. Furthermore, the operational autonomy is assessed in an environment consisting of operational autonomous agents. In the simulation scenarios, a transport agent visits each market place which defines its tour and negotiates in each market place additional load. The tour is repeated by a specified number of rounds. The evaluation settings are defined by numerous parameters; the most important ones are the number of market places in the system, the amount of agents per market place, the number of rounds, and the opponent agents' profiles. For the evaluation experiments with the following settings have been performed:

- Experiments with 5 market places, each consisting of 10 agents, and 100 rounds (5000 negotiations per experiment)
- Experiments with 10 market places, each consisting of 10 agents, and 100 rounds (10000 negotiations per experiment)
- Experiments with 5 market places, each consisting of 10 agents, and 200 rounds (10000 negotiations per experiment)

For each experimental setting, 10 experiments with an operational autonomous (learning) agent and 10 experiments with a non-operational autonomous (non-learning) agent are performed. Variance analysis is applied to statistical analysis, v. The key performance indicator in the statistical model is derived from the summarized learning effect as well as from time. The operational autonomy is tested in four groups: conservative Discourse Agent in aggressive markets (KO), conservative Discourse Agent in conservative markets (OK), and offensive Discourse Agent in mixed markets (OM). These groups are tested in varying configurations with respect to the amount of markets per tour and amount of tours (cf. Table 2.2).

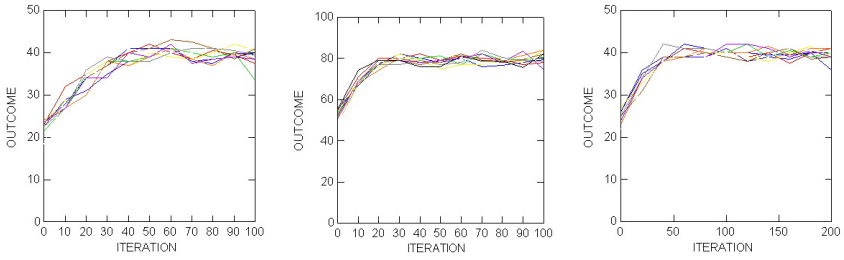
Identifier	Experiments	Amount of Tours	Amount of Markets
KO-5-100	TI1 – TI20	100	5
KO-10-100	TI21 – TI40	100	10
KO-5-200	TI41 – TI60	200	5
KM-5-100	TI81 –TI100	100	5
OK-5-100	TI61 – TI80	100	5
OK-10-100	TI141 – TI160	100	10
OM-5-100	TI101 – TI120	100	5
OM-10-100	TI121 – TI140	100	10

Table 2.2: Configuration of repeated experiments

In the first group of experiments for example, a conservative agents in an aggressive market are evaluated with respect to the operational autonomy on the basis of 250.000 negotiations. There are three experimental settings dealing with conservative agents in aggressive markets. The cumulative means and standard deviations are printed in Table 2.3. The associated success rates are illustrated in Figure 2.3. The results proof, that agents with operational autonomy defined by oac are well capable to adapt their communication structure to new and unforeseen environments even if they differ from their initial environment.

Experiment		Sum		Price		Success		Initial	
		Mean	SD	Mean	SD	Mean	SD	Mean	SD
KO-5-100	Learning	40,27	12,69	1,17	0,18	33,44	6,37	1611	153
KO-5-100	Non-Learning	11,96	0,53	0,69	0,01	17,21	0,75	0	0
KO-10-100	Learning	92,53	20,04	1,26	0,13	72,71	9,46	1684	100
KO-10-100	Non-Learning	23,97	0,84	0,69	0,01	34,53	1,26	0	0
KO-5-200	Learning	41,01	12,84	1,18	0,18	33,78	6,33	1613	157
KO-5-200	Non-Learning	12,12	0,52	0,69	0,01	17,49	0,80	0	

Table 2.3: Cumulated results for groups *KO*



(a) KO-5-100 (b) KO-10-100 (c) KO-5-200

Figure 2.3: Success rate of conservative agents in aggressive markets

The comparison of these operational autonomous resp. LoA-1 with conventional agents following strong regulation i.e. LoA-0 agents is performed on the basis of dif-

ferent key performance indicators. An obvious result from explorative analysis is supported by statistical analysis: the development of the key performance indicators between learning agents is superior to the non-learning agents (cf. Figures 2.4, 2.5, 2.6); these figures indicate a low deviation. These effects have been observed in any of the three experimental groups showing a high reproducibility of the experiments as well as a dependable behavior of the learning algorithm. The variance analysis ANOVA (cf. [Timm, 2004b]) supports this assumption. For all four key performance indicators there are highly significant differences ($p < 0.0001$) between learning (LoA-1) and non-learning agents (LoA-0).

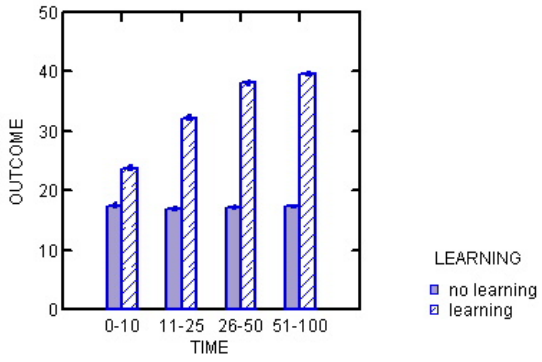


Figure 2.4: Success of learning and non-learning Discourse Agents (KO-5-100)

Additionally, the learning effect over time is also highly significant ($p < 0.0001$, cf. Figures 2.4, 2.5, and 2.6). Finally the interdependencies between these effects are highly significant, as the strongly regulated agent (non-learning) shows no trend of improvement over time while the operational autonomous agent (learning) improves significantly over time.

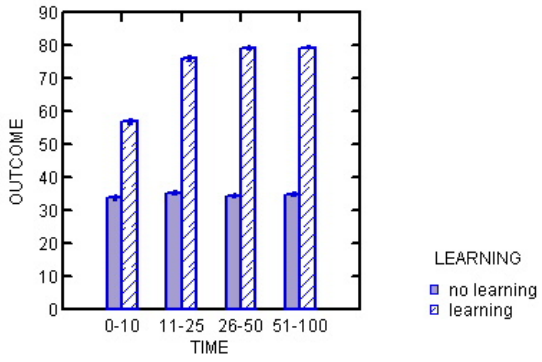


Figure 2.5: Success of learning and non-learning Discourse Agents (KO-10-100)

In consequence, the hypothesis behind this experiments holds for the scenario. For the results of the experiments with the Discourse Agents in further settings, please refer to [Timm, 2003] and the appendix.

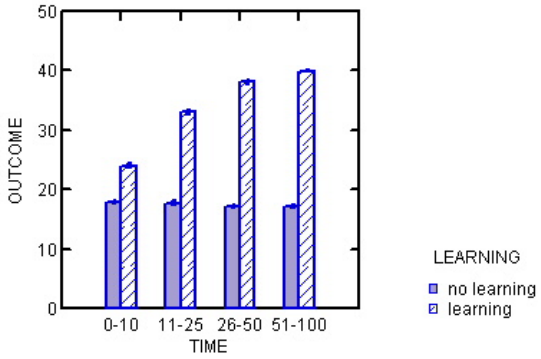


Figure 2.6: Success of learning and non-learning Discourse Agents (KO-5-200)

2.5 Tactical Autonomy

The standard approach to tactical autonomy is to apply a reasoning infrastructure such as the BDI architecture to an agent. This enables an agent to select its intentions from the desires with respect to concrete situations. In our research, we extended this approach of autonomy covering intentions by reasoning on capabilities. Next to goal activation within an agent it is crucial for an agent to decide on its capabilities with respect to the tasks which may not be known at design time. Providing services within multiagent systems, an agent has to register itself with a distinct description of its main capabilities in some kind of yellow page services; following the FIPA standardization, a directory facilitator is used as a service directory [FIPA, 2002]. If another agent requests to solve a specific task, it has to be decided whether or not the requested agent is capable of performing the task successfully. We assume that task requirements as well as capabilities are specified using ontologies. Decision is easy if the concepts of requested task requirements are directly mapped to concepts of provided capabilities. However, concept inequality may occur. Thus, enhanced methods like ontology-based capability management introduced in [Timm and Woelk, 2003b] are established to address this problem. In the following, mutual concepts will be introduced; for further details please refer to [Timm et al., 2006b], [Scholz et al., 2004a], and [Scholz et al., 2004b].

Capability management becomes a key source for cooperative distributed problem solving (CDPS), one of the main features of multiagent systems. Communication of capabilities facilitates a very flexible way of organizing collaborative work among agents [Conen and Neumann, 1998]. The process of CDPS can be divided into three stages: (i) problem decomposition, (ii) sub-problem solution, and (iii) answer synthesis [Smith and Davis, 1980]. An implicit assumption is, that each of the problem solving

agents has knowledge about its capabilities. However, in this approach of tactical autonomy, a more sophisticated solution is required asking: *How does a problem-solving agent knows whether or not it is capable of solving the requested problem?*

The approach to tactical autonomy in this approach is based on the contract net protocol as a common representative for a task decomposition algorithm [Smith, 1988]. The contract net can be described step-wise: in each step, problems are broadcasted to any problem solving agent, these agents provide a complete solution by itself or it decomposes the task. For the next step, it changes the role and requests solutions for each sub-task. Of course, a problem solving agent has to be able to decide whether it has the capability to solve a specified task. In real-world applications, there are methods to solve a problem, which are not equal but equivalent. E.g. drilling is the standard operation for creating a hole in a work piece, but in some cases and with specific machine tools milling can produce an equivalent result. Nevertheless, both actions are not equal. In this context, the mapping of capabilities to tasks and vice versa, which is not based on simple concept or string equality is an open issue.

In the definition of local states, we introduced desires as (multi-)modal formulas which are satisfied in some future state. Transferring this concept to the specification of problems, problems are formulas, which describe pre and post conditions. We assume that an explicit formal ontology is provided for the specification of the semantics of used symbols in these conditions. There are multiple approaches to formally define an ontology. In this context, we use a minimal definition of ontologies following the assumption that an ontology has at least a set of concepts and a taxonomic relation.

6 DEFINITION (ONTOLOGY)
Let C be a set of concepts and r^t a taxonomic relation then a tuple
$Onto = \langle C, r^t \rangle$
is called ontology.

As abbreviation we use *length* as a symbol for the minimal amount of edges between two nodes, $length(n)$ for the length of the shortest path from the root node to node n, and $length(n, m)$ for the length of the shortest path between node n and node m. The mapping $mscc$ maps a pair of concepts to the most specific common concept.

The capability management to enable tactical autonomy is based on the setting, that the agent receives a problem description. Then, the agent uses its ontology as well as the explicitly specified capabilities and performs the cobac* algorithm to decide how appropriate the agents capabilities are for the specified problem. Following the definition of Discourse Agents, capabilities are determined by a set of plans. Each action plan describes a single capability as a transformation from a state which satisfies its pre condition into a state which satisfies its post condition. Taking the taxonomic relation from the ontology into account, the agent can extend its capabilities by subsumption. The approach of capability management extends not only capabilities but also problem descriptions. If a problem description uses a specific concept and the capability of an agent matches to the super-concept, the capability may be sufficient. The cobac* approach introduced in [Timm et al., 2006b] is based on the general procedure of the conflict based agent control (cobac) and will be introduced briefly in the next section (cf. Section 2.6). The underlying idea of cobac is to evaluate conflict and synergy

between two goals which is based on partial correspondence and contrast of complex formulas. In opposite to this, the cobac* algorithm introduces a conflict measure based on the evaluation of proximity and distance in a taxonomic relation of an ontology.

The algorithm aims at identifying that capability which fits *best* to the requested problem. The formal representation of a problem is defined as a tuple of pre (starting state) and post conditions (goal state), i.e., $problem = \langle \varphi^p_{pre}, \varphi^p_{post} \rangle$. cobac is based on conflict and synergy assessment and the optimization problem of minimizing conflict and maximizing synergy. The same principle is applied to the capability management process. Here, the compatibility of capability and problem is in question; the optimization should reach a solution where the level of compliance and diversity. The resulting algorithm consists of the following steps:

1. Option generation,
2. Problem-oriented conflict and synergy assessment,
3. Option selection.

The algorithm starts with creation of options, i.e., identification of capabilities which could be used for solving at least a minimal aspect of the problem. In context of the formal definition, cobac* checks the available capabilities of the agent, i.e., the capabilities, which pre conditions are satisfied in the current state.

7 DEFINITION (OPTIONGENERATION)
A set of options Opt is constructed for a problem $\langle \varphi^p_{pre}, \varphi^p_{post} \rangle$ as specified by:
$Opt = \{\langle \varphi_{pre}, \varphi_{post}, A, st, se \rangle | \varphi_{pre} \rightarrow \varphi^p_{pre}$ *and* $\langle \varphi_{pre}, \varphi_{post}, A, st, se \rangle \in Plan\}$

In the next step proximity and distance between options and problem are assessed. Therefore, each option i is put into relation to the problem and the distance and proximity values are calculated for each pair of concepts in $\varphi^p_{post}, \varphi^i_{post}$. The distance $\kappa(p, o_i)$ and proximity $\sigma(p, o_i)$ potential is computed for each option as the sum of all of its distance and proximity values.

8 DEFINITION (PROXIMITY AND DISTANCE)
For each option $o_i \in Opt$ and problem $p = \langle \varphi_{pre}, \varphi_{post} \rangle$ with a taxonomic relation r^t of the ontology, the distance κ and proximity σ potential are calculated as follows:

$$\kappa(p, o_i) = \sum_{c_i, c_p \in C^t \wedge c_i \in \varphi^p_{post} \wedge c_p \in \varphi^i_{post}} k(c_p, c_i) \text{ (distance) and}$$

$$\sigma(p, o_i) = \sum_{c_i, c_p \in C^t \wedge c_i \in \varphi^p_{post} \wedge c_p \in \varphi^i_{post}} s(c_p, c_i) \text{ (proximity).}$$

The proximity and distance values of a tuple of concepts c_i, c_j are determined by the length in the taxonomic relation:
$k(c_i, c_j) = length(c_i, c_j)$ *and*
$s(c_i, c_j) = length(mscc(c_i, c_j))$.

Finally, the option selection step is applied to the proximity and distance potentials for identification of the *most* appropriate option for the requested problem. Therefore, the quotient is computed as follows: $\psi(p, o_i) = \frac{\kappa(p,o_i)}{\sigma(p,o_i)}$. The integrated proximity and distance potential is minimal if there is no distance between the concepts and maximal (resp. infinite) if no proximity could be identified. The option with the lowest quotient ψ is used as resulting capability, which is proposed as solution to the task agent. If this quotient is zero, the proposed problem solving capability is equal to the requested problem.

In the approach introduced in our research, e.g., [Timm et al., 2006b], is focused on tactical autonomy of a problem solving agent. The perspective of an agent, which requests a solution is not in the focus of our research. For a requesting agent, the tactical autonomy includes the decision on the question, if a proposal for an approximate solution should be accepted. We propose to use the conflict-based agent control cobac algorithm (cf. Section 2.6) for identifying valid solutions with respect to its desire and the adaptive communication oac (cf. Section 2.4) for cost bargaining if the solution is only similar and not equal to the requested problem.

Alternatively, cobac* can be used to build an equivalence relation within the ontology: A simplified approach uses the taxonomic relation of the ontology and a maximum distance value mdv. This value determines if a solution provided with the capability c_i is valid for the problem c_p: $(length(c_i, c_p) < mdv)$.

2.6 Strategic Autonomy

Agents can be used as ”enterprise delegates” for supporting tasks like the management and integration of planning, scheduling, and controlling processes. The approaches introduced in the last two sections, enable agents to act autonomously with respect to operational and tactical decisions. However, the agent is still bounded by the desires which are pre-defined by the user and may not be adapted easily following the conventional BDI approach. The implementation of real-world applications requires the definition of objectives and the instantiation of algorithms for goal creation. Strategic components may be introduced into BDI architecture if previously contradictive general goals are allowed. The strategic task of the agent is to create new goals as strategic compromises meeting the contradictive set of goals as well as possible and to follow these new goals as strategic concepts in its decisions. Thus conflict resolution is the key to strategic autonomy of agents.

A new approach was introduced to the identification of synergy and conflicts of interest in the desires as well as the intentions. On this basis, a conflict resolution mechanism is applied for minimizing conflicts and maximizing synergy. In this step, new goals as compromises of desires might be created. The algorithm, formal specification as well as an extensive evaluation are provided within the paper proving the beneficial behavior of an explicit conflict-based selection process. More details on this approach have been published in [Lorenzen et al., 2006], [Timm, 2004b], [Timm, 2001b], [Herzog et al., 2001], and [Timm et al., 2001a].

The main decision function is $decide$. If this function is computed, the following four steps are processed:

1. Intention reconsideration,
2. Option generation and assessment,
3. Conflict management and resolution, and
4. Option selection.

In the first step, current intentions of the agent are reconsidered. The result of intention reconsideration is a revised set of intentions, which does not include intentions, which do not fit the criteria of the pre-defined level of commitment (blind, single-minded or open-minded commitment) [Rao and Georgeff, 1991]. In the deliberation process an option is defined as a tuple of a desire and a plan (cf. intention, $o = \langle des, plan \rangle$, with $des \in D$ and $plan \in Pln$). The option generation process builds a set of options O, which contains the complete set of intentions I and new options. New option is created for a desire, if no intention is pursuing it and the desire is accessible. The accessibility relation is defined in consideration of the branching temporal structure [Wooldridge, 2000]; modifications to this relation are done with respect to decidability and efficiency. During the creation of a new option a plan is selected for pursuing the desire in question using a plan allocation function.

An evaluation function is assessing each option of the option set O, using the desire assessing function γ and the current state of the plan, such that an option with an almost completed plan will receive high priority within the option filtering process. Next to the intention reconsideration, this evaluation function implements the commitment to an intention and should ensure that important and almost completed tasks will be finished first.

In the next step the options will be filtered. The filtering uses conflict assessment and resolution introduced with the Discourse Agent architecture. For each pair of options, a synergy as well as a conflict value is calculated. Two options receive a high synergy value if they are pursuing similar desires and the plans are not contradictory, e.g., the post condition of plan A is not prohibiting the pre condition of plan B. Figures 2.7 and 2.8 show the conceptual idea of synergy and conflict.

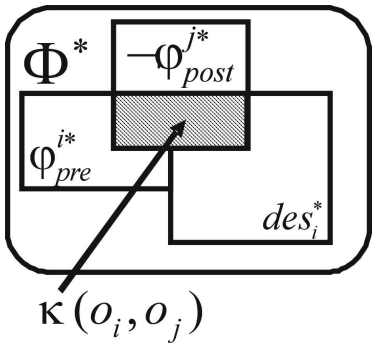


Figure 2.7: Conflict value

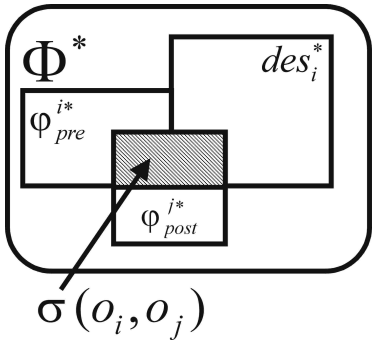


Figure 2.8: Synergy value

A conflict and synergy potential is calculated as the sum of each conflict and synergy value and used as a performance indicator within the process of conflict resolution. The conflict classification and resolution algorithm is motivated from the field of interpersonal conflict studies [van de Vliert, 1997]. A conflict taxonomy is introduced in [Timm, 2001b], where each pair of options is classified as a leaf in this taxonomy (cf. Figure 2.9, for details refer to [Timm, 2004b]). For each type of leaf, there is a resolution strategy taking the cooperation or conflict potential into account, e.g., if two objectives are very similar, they can be merged in a cooperative setting.

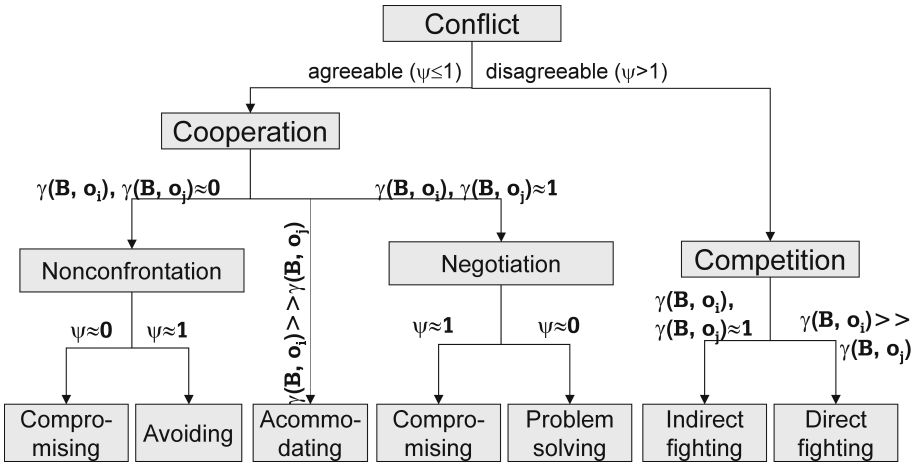


Figure 2.9: Conflict classification scheme

The last step of the $decide$-function is to filter the options to create a new set of intentions. Thus, each option must meet a minimum evaluation value to be treated as an intention. The $decide$-function is formally defined as follows:

9 DEFINITION (DECIDE) *The mapping* $decide : \mathcal{L} \rightarrow \mathcal{L}$
with $decide(\langle B, D, I, Pln, \gamma \rangle) \mapsto \langle B, D, I^*, Pln, \gamma^* \rangle$ *is called conflict-based agent control iff*

- $irf : \langle B, I \rangle \mapsto I^0$ *is an intention reconsideration function,*
- $go : \langle B, D, I^0, Pln, \gamma \rangle \mapsto O$ *is an option generation function,*
- $crf : \langle \langle B, D, I^0, Pln, \gamma \rangle, O \rangle \mapsto \langle \langle B, D, I^0, Pln, \gamma^* \rangle, O^* \rangle$ *is a conflict resolution function, and*
- $filter : O^* \mapsto I^*$ *is an option filtering function.*

Evaluation

The evaluation of the cobac algorithm is performed in the IntaPS scenario (cf. [Timm et al., 2004], [Lorenzen et al., 2006], Section 5.1). The agent which uses the cobac resp. a comparison algorithm represents a machine tool (resource) within a shop floor. Here, order agents representing manufacturing orders negotiate with resource agents to implement an integrated process planning and production control system. In this setting, the resource agent is responsible for its own schedule as well as the negotiation with the order agents. There is an explicit probabilistic failure model included for the resources. In order to assess the behavior of the cobac algorithm, an evaluation with 1700 experiments and 163.200 decision cycles has been performed. Additionally, a state-of-the-art intention selection algorithm based on dynamic priorities has been implemented in order to benchmark the cobac algorithm. While the cobac algorithm represents an approach to enabling strategic autonomy, the priority-based comparison algorithm implements tactical autonomy with simplified BDI reasoning. The cobac algorithm as well as the priority intention selection algorithm are each tested with a set of generic desires (five desires) and an extended set of partially composed desires (seven desires). The underlying branching temporal structure of the cobac algorithm is analyzed within the evaluation, too. Different computing modes for the accessibility relation for desires have been implemented varying from constant over linear to exponential computational time usage. The experiments have been used to assess the behavior of the cobac algorithm in different situations as well as to benchmark it against the priority intention selection.

The experiment groups are varying with respect to the amount of goals (five or seven), the computation of the temporal structure (n.a., simplified structure, restricted structure), and the decision mechanism (reference point, cobac, priority). The resulting experiment groups are defined as follows:

- *No computation of branching temporal structure:*
  reference point (IR), five desires (cobac: ISCN, priority: ISPN)
  seven desires (cobac: IXCN, priority: IXPN)

- *Simplified computation of branching temporal structure:*
  five desires (cobac: ISCL, priority: ISPL)
  seven desires (cobac: IXCL, priority: IXPL)

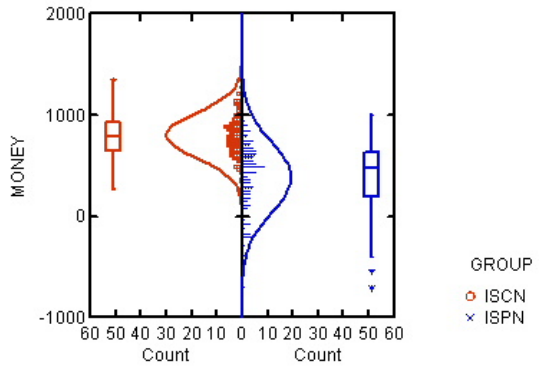
- *Restricted computation of branching temporal structure:*
  five desires (cobac: ISCC, priority: ISPC)

- *Varying configurations (increased failure rate):*
  reference point (I2R), five desires (cobac: I2SCN, priority: I2SPN)

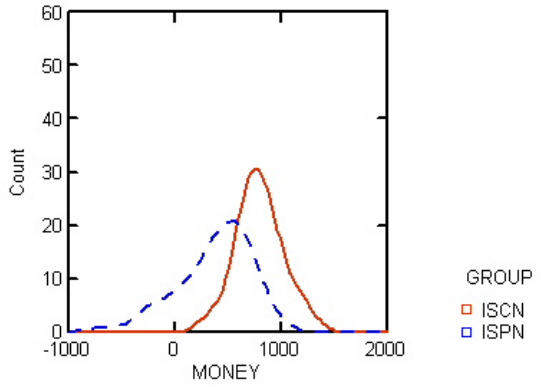
- *Varying configurations (reduced failure rate):*
  reference point (I3R), five desires (cobac: I3SCN, priority: I3SPN)

These experiments are used to analyze the behavior of the cobac algorithm including its limitations compared with an alternative algorithm. The hypothesis is, that the

cobac algorithm is beneficial in comparison to priority based intention selection. Furthermore, the question arises, if the design and structure of desires has impact on the quality of an agent's behavior. Thus, the experiments with seven desires include desires which are compromises of the other five desires. Here, the benefit of cobac should be significantly decreased. For an overall performance evaluation, an almost optimal solution has been implemented, which is a static solution specialized for this setting using internal information on error probability rates etc.. Further experiments are dedicated to the analysis of limitations of the approach (variation of error probability rate).



(a)



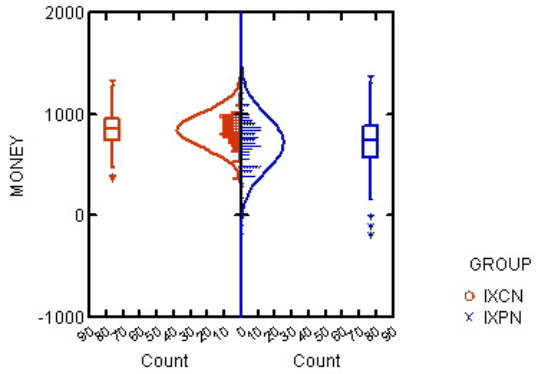
(b)

Figure 2.10: Statistical test for money indicator (ISCN versus ISPN)

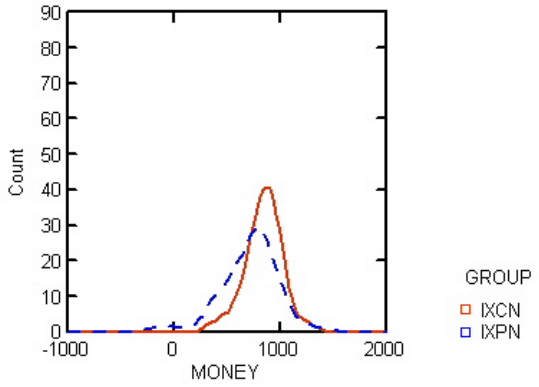
In direct comparison of a cobac based and the priority-based agent, cobac is superior in any of the five experiment groups (cf. [Timm, 2004b]). In the case of five desires, the distance between cobac and priorities is even higher, i.e., cobac gains more than twice the amount with respect to the money performance indicator (cf. Figure 2.10). This distance decreases – as expected – if results of the cobac algorithm are integrated into the desire set (cf. Figure 2.11). The benefit of cobac is based on the

ability to dynamically generate new intentions as combination resp. compromise of existing desires. The priority-controlled agent shows a significant statistical spread in the success parameters, especially the number of accepted but not processed orders is a major problem for the priority-based agent, since it fails to handle the backlog when it reaches a certain amount. The Discourse Agent using the cobac algorithm appears to be more stable and overall superior to the priority-based agent.

The benefit of cobac is supported by the statistical analysis where the deviation of cobac to priorities is highly significant ($p < 0.0001$) with respect to the indicators money, maintenance level, production, and order list. However, there is an exemption: the maintenance level is not significant if the branching temporal structure is computed ($p = 0.1006$ resp. $p = 0.1063$ by Kruskal-Wallis-Test).



(a)

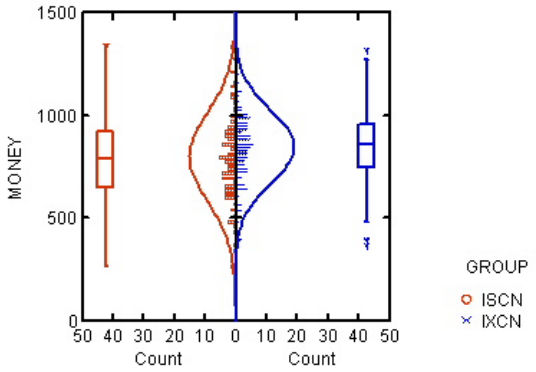


(b)

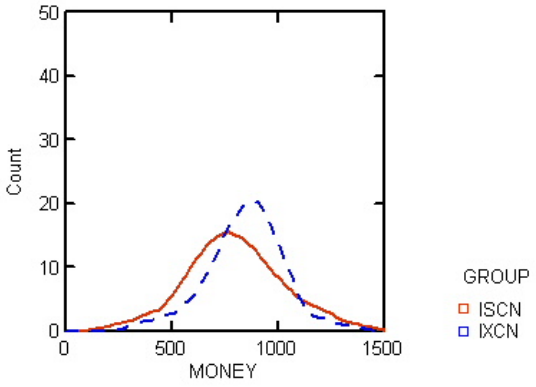
Figure 2.11: Statistical test of indicator money (IXCN versus IXPN)

For the evaluation of the cobac approach it is also important to consider the influence of desire sets to the performance of the algorithm as well as to the comparison between priorities and cobac. The results of those experiments, where no branching

temporal structure was computed, is highly significant proving the benefit of a larger desire set with pre-combined desires with respect to the order list ($p < 0.0001$). For the money indicator, there is also a significant benefit of the larger desire set ($p < 0.0001$ for the *t*-Test and $p = 0.0314$ for the Kruskal-Wallis-Test, cf. 2.12). If a branching temporal structure is computed, there is not that clear result. However, for the order list the results are still highly significant for the larger desire set ($p < 0.0001$).



(a)



(b)

Figure 2.12: Statistical test of indicator money (ISCN versus IXCN)

An interesting side effect of this statistical analysis is that there is no proven benefit of computing complex temporal structures with respect to statistical significance. In any of the experiment groups with varying temporal structures (e.g., Figure 2.13) there are undetermined effects to this result. Nevertheless, there is a problem with computing the restricted branching temporal structure: caused by the exponential memory usage, the calculation was restricted to four steps into the future. There is a surprising maleficent behavior here. However, as this question has not been regarded in the experiment design, these results are of exploratory value only but may constitute a starting point

for more research on this topic.

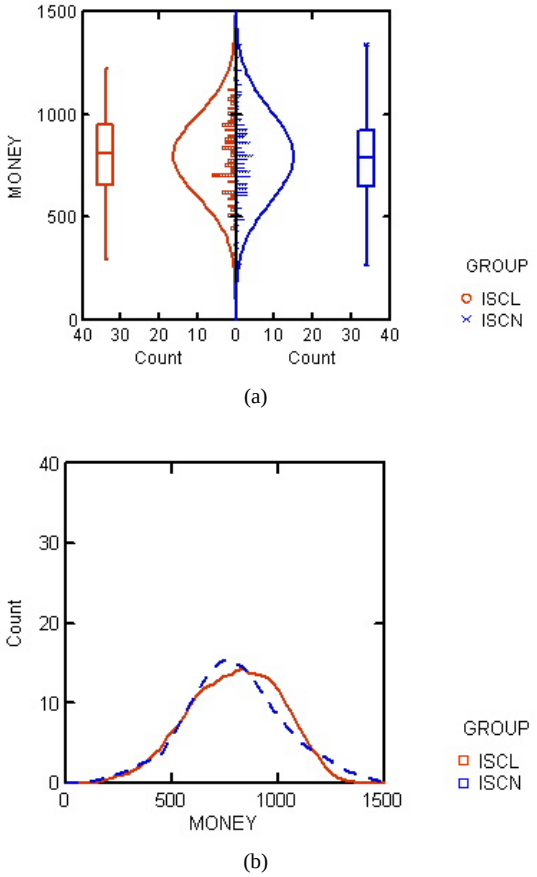
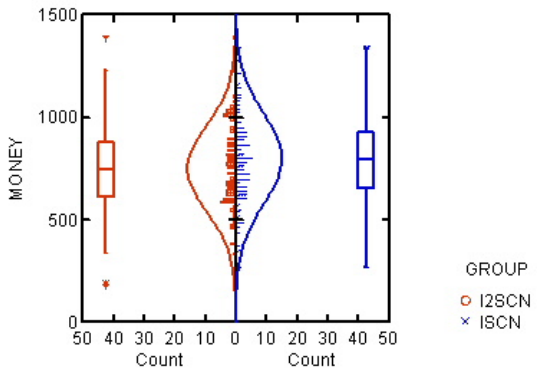


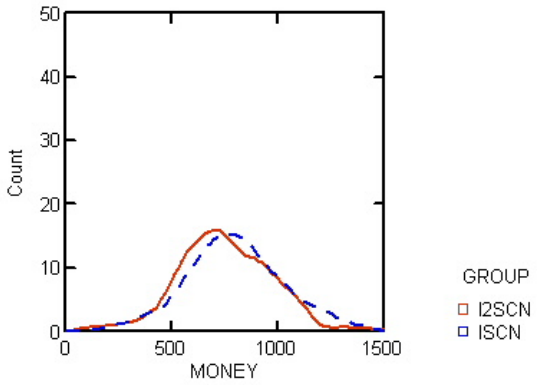
Figure 2.13: Statistical test of indicator money (ISCN versus ISCL)

For the evaluation of the boundaries and dependability of the cobac approach, the scenario was varied with respect to the failure probability: The experiments in the group I2 have been performed with an increased failure probability ((1.00) while the experiments in the group I3 have been simulated with a decreased failure probability (0.05). The results indicate, that cobac compensates this effect adequately, i.e., there is no significant deviation between the experimental groups in the money and order list indicators (e.g., Figure 2.14). However, there is a significant deviation between the settings with respect to the maintenance level in the reduced failure probability setting. Here, the benefit of reduced failures supports a better performance with respect to the order list (cf. [Timm, 2004b]). The comparison between cobac and priority-based intention selection is similar to the original scenario, i.e., the superiority of cobac is significant for the maintenance level indicator and highly significant for the other indicators.

Finally, the cobac algorithm has been compared to an algorithm designed specifically for this scenario. in The results of this algorithm can be considered as local maxima. In table 2.4, the results of cobac and the priority-based algorithm are listed in percent with respect to the local maxima.



(a)



(b)

Figure 2.14: Statistical test of indicator money (ISCN versus I2SCN)

It is obvious, that the cobac achieves about 70% in average while the priority-based algorithm results is less than 30% of the reference point when considering the small desire set. This changes for the larger desire set; however, there is still a benefit between the cobac and the priority approach. The main problem indicated by these results is the size of the order list. The order list with the cobac algorithm is doubled to tripled in comparison to the reference point. These results support the effects identified by the statistical analysis mentioned before. For further details on the experiments, statistical exploration, and statistical analysis, please, refer to [Timm, 2003].

Experiment Group	Money	Maintenance	Production	External Buy	Order List
IR	100,00	100,00	100,00	100,00	100,00
ISCN	74,03	98,85	38,80	112,00	198,76
ISPN	35,46	88,61	5,16	107,14	591,66
ISCL	74,10	97,26	37,52	112,84	201,74
ISPL	30,52	90,16	5,91	104,84	625,68
ISCC	70,60	96,15	38,68	111,93	206,35
ISPC	33,69	89,74	5,38	105,53	608,32
IXCN	78,59	97,30	47,25	110,83	162,12
IXPN	66,81	103,21	17,13	113,93	303,26
IXCL	76,70	97,05	48,60	110,84	160,68
IXPL	64,38	103,16	17,76	113,93	284,92
I2R	100,00	100,00	100,00	100,00	100,00
I2SCN	75,23	98,32	38,09	109,01	222,31
I2SPN	39,72	88,31	4,37	104,05	579,29
I3R	100,00	100,00	100,00	100,00	100,00
I3SCN	68,72	95,41	37,99	113,44	172,81
I3SPN	31,07	88,37	6,80	107,21	636,87

Table 2.4: Reference point in comparison to cobac and priority

Chapter 3

Strategic Management of Multiagent Systems

As introduced before, intelligent agents are an adequate means for implementation of autonomous software systems on the respective level of autonomy. However, intelligent agents are assumed to establish an emergent effect within groups or systems of agents, the so-called multiagent systems. The question arises, if such an emergent behavior, i.e., a macroscopic behavior on the basis of microscopic interactions, is beneficial for the global system. In the beginning of multiagent research, this assumption was stated as a fact. Recent research focuses on sophisticated design of the autonomous subsystems to enable a positive effect of the whole system [Liu, 2001], [Liang and Zhu, 2005]. [de Wolf and Holvoet, 2005] propose an approach for engineering self-organizing systems. Their approach is based on the analysis of the system after implementation and before delivery. Because of the well-known complexity of testing concurrent systems, the approach seems to be adequate for systems with a moderate amount of internal states, where no extensive internal states or static strategic behavior exists.

In this chapter, we introduce roles and structures for multiagent systems as a starting point for strategic management. For the adaptation to concrete situations, we propose a versatile management for the dynamic reorganization of multiagent systems. However, this approach does not consider the effects of strategic management to individual agents. Thus, we discuss an approach for reflection. The autonomy of individual agents is considered with respect to the multiagent systems performance. In critical system states, the autonomy of the agents is adjusted.

3.1 Roles and Structures in Multiagent Systems

So far we discussed single autonomous systems. In context of the strategic management, we focus on whole systems consisting of autonomous subsystems. This research has a strong analogy to multiagent research. A conventional approach to establish

some kind of management in the system is the introduction of roles and structures in the system.

The implementation of strategic, tactical, or operational objectives in autonomous systems requires the interaction of the underlying autonomous subsystems. These goals can for example be addressed to shared information, coordination, or flexible conflict resolution. [Ferber, 1999] introduced interaction as a core functionality for emergent organization of multiagent systems. Therefore he identifies a set of assumptions: the presence of agents capable of acting and/or communicating, constructs which can serve as a meeting point for agents, and dynamic elements allowing for local and temporary relationships between agents [Ferber, 1999].

On an operational level, agents interact on the basis of sequences of messages, e.g., protocols or oac plans. Communication on this level is very flexible for coordination. However, the organization of the agents resp. groups has to be negotiated for any problem solving process again and cannot be reused easily. Hence, there could be a significant overhead for coordination in the initialization step between agents leading to high communication costs and complexity. For the tactical autonomy in systems of autonomous subsystems it should be assumed that their beneficial organization patterns could be reused in further coordination processes. Doing so, a mid-term behavior could be established in subsystem organization. In economics and organization theory, there are two main concepts to specify actors in an organization: structure and roles [Kieser and Kubicek, 1992]. In DAI research, these concepts have been transferred from economics to multiagent research [Kirn, 1996]. With respect to autonomous systems, the concept of roles is restricted to the representation of cognitive states as the basis of permissions or responsibilities. Therefore, these concepts can be used for the implementation of organizational views on internal representations of even huge world models, which allows for efficient inferences [Timm et al., 2006d]. Recent approaches towards the design of open multiagent systems are explicitly modeling social models including roles, which are not limited to cognitive states [Petsch, 2002]. The concept of roles focuses on different views on individuals and realizes an abstraction of individual abilities, goals and behaviors. Due to external expectations on roles, communication efforts should be reduced significantly. Roles can be beneficially applied to the capability management for implementing tactical autonomy in autonomous subsystems.

Inter-role conflicts still have to be solved by communication in each interaction again. To establish long-term cooperations as strategic perspective in a system of autonomous subsystems, we propose the introduction of structures. For organizing large scale systems, structures can be used to establish reusable cooperation patterns. In real-world organizations, there are specific concepts for individuals joining or leaving structures with respect to task classes, tasks, and capabilities [Hill et al., 1998]. Organizational structures are abstracting from a group of individuals to an external representation which group details. Therefore, hierarchic autonomous systems can be engineered where an autonomous system is implemented by a group of autonomous systems, e.g., an enterprise participating in a supply chain can be represented by its buying department. Further details on the application of roles and structures in multiagent systems as well as properties of multiagent systems are provided in [Timm et al., 2006d].

3.2 Versatile Management for Multiagent Systems

The conventional approach to the design and implementation of a software system is to specify the system as a whole or to identify clear interfaces and modules at design time. In autonomous software systems with strategic autonomy, we assume that the configuration resp. organization of the system is part of the runtime behavior. Each software system has a principal, i.e., an owner or organization, which has deployed the system or is responsible for its behavior. In systems of autonomous subsystems, however, the subsystems can belong to different principals. Software systems should follow the strategic perspective of their respective principals. In our research, we outlined an approach on versatile management for a pre-organized system of autonomous entities in the manufacturing domain. The approach allows the system of autonomous subsystem to identify change indicators and initiate re-organization accordingly. This approach is closely related to our research on emergent virtual enterprises [Tönshoff et al., 2002a]. This approach can be generalized with respect to strategic management of autonomous software systems. For more details on agent-based versatile management, please, refer to [Scholz et al., 2005a].

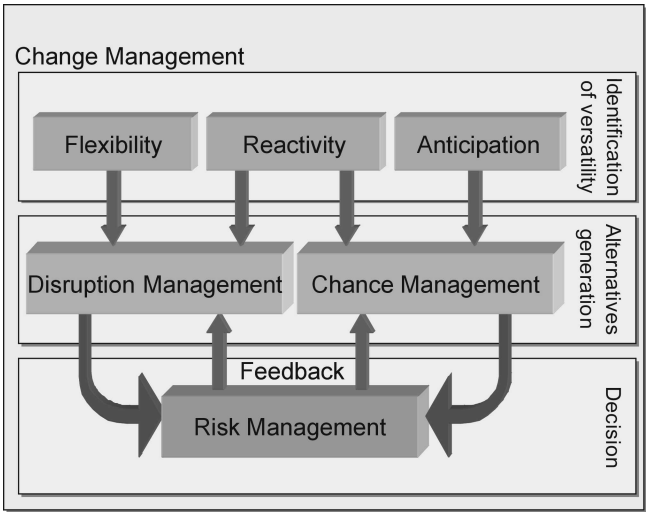


Figure 3.1: Architecture for change management

In the manufacturing domain, versatility is a key challenge for competitive enterprises. The versatility in focus of our research is addressed to the structural reorganization of shop floors. Transferred to strategic management of systems of autonomous subsystems, organizational structures of autonomous systems should be reorganized with respect to changes in the environment. The approach is based on indicators for organizational changes. The key challenge is to identify situations, where a change of the structure is indicated. Reorganization of the structural system is classified as first and second order change. First order changes are adaptations of the system structure with respect to minor changes in the environment. The modification of the structural

system is based on observations of the environment and can be part of a continuous improvement process. If the minor changes are not sufficient for the improvement of the system, a second order change takes place. Here, the system structure is reorganized in a ”revolutionary” way.

The research on versatile management in the manufacturing domain, lead to an architecture for change management of autonomous systems as conceptualized in Figure 3.1 [Scholz et al., 2005a]. In the following, we will introduce the abstract architecture and shortly describe the required components. Change management is based on a three layer architecture: Identification layer, generation layer and decision layer.

The change management architecture enables a system to initiate a change for two purposes: flexibility and reactivity. Information on change in regard to flexibility includes factors for spontaneous disruptions in the normal operation of the system. Disruptions, e.g., traffic jams in logistics, lead to minor adaptations of the system following the first order changes. Generally, The second component on the identification layer, reactivity, is addressed to problems which require reorganizations with respect to changes in the environment (first order change). Additionally, tactical aspects are evaluated in this component, i.e., the mid-term adaptation or the tactical autonomy of systems of autonomous subsystems can be implemented by this component. For long-term adaptation of the system, the anticipatory component is applied. This component is activated if the system structure prevents efficient system behavior, for example new resources, i.e., autonomous subsystems, enter the system and cannot easily be integrated in the system structure. This component enables the system to take upcoming chances and therefore is part of second order change.

After the identification of change potential, the second layer is initialized. Here, the system generates alternatives to the current structure with respect to first or second order change. In the case of disruption as a change indicator, the system generates options to cope with the disruption in a reorganized way. In the other case, chance detection as change indicator, the chance management is initialized, which computes possible reorganizations under consideration of the emerged possibilities. In chance management, the decision for a newly generated option is enriched by feedback acquired from the decision component. Thus, the system is enabled to react more quickly when similar disruptions occur. The generation of alternatives is aimed to maximize the identified improvement potentials and results in a set of possible change measures. Analogously to disruption management, feedback on the decisions for or against alternatives may be received automatically.

On the basis of the alternatives generated in the second layer, the decision layer has to decide which alternative should be used for a change of the system. For sophisticated evaluation of alternatives, a risk management is included. The context-dependent assessment of risk versus chances is computed for any alternative resulting in a weighted list of alternatives. After a decision has been made, a feedback loop allows for an adaptive change management.

The change management introduced so far is by itself in need of implementations for strategic autonomy, for decision making on change alternatives, tactical autonomy, for identifying different alternatives on the basis of capability management, and operational autonomy to cope with minor disturbances in the application without initiating a change immediately.

3.3 Reflection in Multiagent Systems

Multiagent systems consist of autonomous agents, which interact on a local – microscopic – level. Optimal macroscopic behavior does not necessarily emerge from those interactions. As discussed in section 1.1 microscopic optimization can lead to local optimal situations. However, a key benefit of multiagent systems is assumed to be a positive emergence on a system level. In social science, the phenomena of microscopic-macroscopic interaction is widely researched [Schegloff, 1987], [Bohman, 1993], [Alexander and Giesen, 1987]. Norms and regulations are introduced in a social system to establish a better system performance. The research outlined in this section is focused on the question, if results from social science can be fruitfully applied to multiagent systems. Therefore, we consider the theory on reflection introduced by [Luhmann, 1984]. In this work, a system has the ability to reflect about its overall performance explicitly within negotiations. In an interdisciplinary research, [Timm and Hillebrandt, 2006] conceptualized a social mechanism as an explanatory model for societies based on the work of [Luhmann, 1984]. On this basis, a conceptual model for reflection in multiagent systems has been developed.

The multiagent conceptualization of reflection is based on the assumption that a multiagent system was chosen deliberately as a system design. In consequence, autonomy of the agents is not a side effect but one of the key features. If dependability on the multiagent system is in question, then some dynamic mechanism is required, which allows for context-dependent adjustment of the individual agents' autonomy. The framework for reflection in multiagent system consists of four stages as illustrated in figure 3.2.

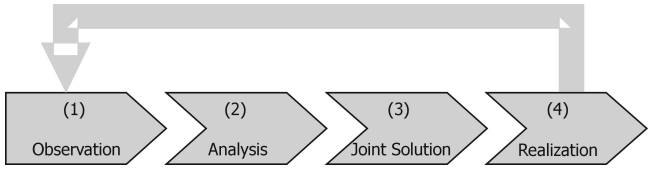


Figure 3.2: Reflection in multiagent systems

The approach is based on a group of agents with operational, tactical, and strategic autonomy as a core element. This group can be formed dynamically in runtime or specified at design-time. In either way, it is assumed, that the group of agents have some common goals and the fulfilment of this goal can be measured the group of agents. Furthermore, for each goal, there are different levels of goal satisfaction, i.e., 0 implies that a goal is completely unsatisfied and 1 indicates that the goal has been satisfied. For utility-based goals a continuous scale is assumed while for logic-based goals the goal-satisfaction is a binary function. Furthermore, we assume that the consideration of global goals in every deliberation step would be inadequate with respect to computational or memory consumption.

In the observation stage, each agent reports its performance to a blackboard or central entity (group coordinator) within the group. The blackboard resp. the group coordinator computes the goal satisfaction on the basis of the individual results. To-

gether with the concrete satisfaction level, the goal satisfaction is broadcasted to resp. available for any agent of the group. If the goal-satisfaction is classified as deficient, the agents should adjust their operational autonomy. Doing so, the agents should plan the next action resp. action sequence under consideration of the global goal. E.g., assume that a BDI agent has instantiated an intention and associates this intention with a partial global plan. The agent would now chose an linearization of the global plan, which is most suitable for supporting the global goal.

The observation stage is used for normal system performance. If the system performance with respect to a specific goal is critical, the multiagent system's state changes to the analysis stage. In this stage, the agents have to communicate their currently pursued goals. The analysis is performed by the agents cooperatively or by a central entity resp. group manager. We assume, that the implementation of a group manager is adequate for the following stages. Here, the group manager has to identify the interdependencies of goal selection of the individual agents and missing system performance on the group level. These interdependencies are published. Under consideration of the autonomy of individual agents, the tactical autonomy has to be adjusted by the agents. Each agent should consider the effects of its goal instantiation, e.g., in our example the step of associating a plan to intentions, with respect to the group performance.

In the case of a severe system performance, the group of agents is transformed into the joint solution group. Here, the group manager mediates the negotiation about individual agents goals. The agents are assumed to improve their strategic autonomy, i.e., the agents instantiate those goals, which help the group performance. This step is part of recent research and has to be elaborated in more detail.

The solution, which has been negotiated in the group and which restored systems performance, is generalized as a social rule for later usage in severe situations (stage four). This approach has been worked out in more detail in [Timm and Hillebrandt, 2006].

3.4 Strategic Control

In section 3.1, we introduced roles and structures for implementation of a basic tactical and strategic autonomy in systems of autonomous subsystems. The change management approach introduced in 3.2 is dedicated to the autonomous adaptation of structures within autonomous systems. In this section we propose an approach for a sophisticated use of roles in systems with autonomous subsystems of different provider, principals, or owners. Generally, roles can be considered as a collection of services [Kirn, 2002]. In open environments, e.g., the internet, autonomous systems provide services by exploitation of service descriptions The key challenge for the strategic control in the context of open autonomous systems is to decide on the quality of service provided by other autonomous systems. In [Scholz et al., 2005b] and [Scholz et al., 2005c], an agent-based approach was introduced for third-party quality of service certification enabling reliable distributed problem solving which has been evaluated by a prototypical implementation. Evaluating quality of service is a challenge by itself; in systems with a high dynamics, it is crucial to evaluate the quality of service dynamically at runtime.

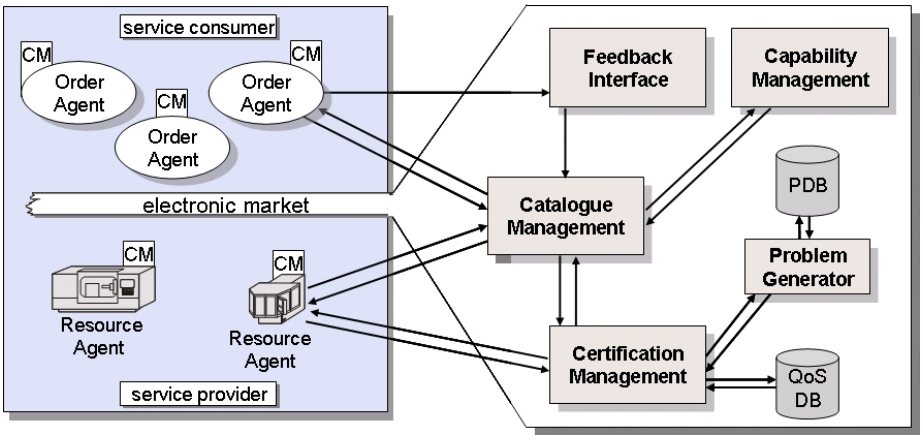


Figure 3.3: Architecture for the dynamic certification of services

In [Scholz et al., 2005c] we propose a conceptual framework for certification management in open systems. We assume, that there is no authority to guarantee the quality of a provided service. Therefore, our conceptual framework integrates identification, evaluation, and selection of services for the reliable behavior of autonomous software systems. The conceptual framework consists of three main components:

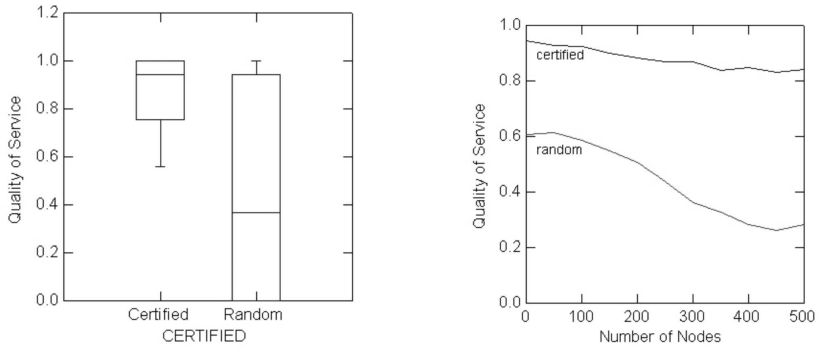
- Capability management for the identification of roles, which are adequate for problem solution. The underlying assumption is that roles are defined in the notion of capabilities [Timm et al., 2006b].
- Certification management for dynamic evaluation of roles.
- Catalogue management as an integrating infrastructure of capability and certification management.

Additionally, the conceptual framework includes interaction protocols for retrieving a role and certification of roles. In order to enable a reliable behavior, the framework requires an autonomous system to certify its roles resp. capabilities prior to registration. As quality of service is not only determined by objective measures, the framework allows for user feedback. In Figure 3.3, the architecture of the certification management system is outlined.

Certification of roles is structured as follows: a set of problems as well as a standard solution is generated. However, problems are domain specific, and are either created autonomously or taken from the problem database (PDB). The problems are communicated to the autonomous subsystem under certification and are solved by this subsystem. In the next step, the certification management receives the solutions from the subsystem and evaluates the quality of solution resp. service. The resulting assessment of the service is stored in the catalogue management resp. the quality of service database. For further details on our specification and implementation of certification management, please refer to [Scholz et al., 2005c].

Evaluation

The approach to certification management was evaluated in an exemplary scenario. The implementation is based on a multiagent system. The scenarios are based on route planning as a capability of the agents. The agents, which should be certified by this approach, are implementing one search algorithm each, i.e., the greedy search, breath-first search, depth-first search, and iterated depth-first search agents were implemented. For the certification step, a problem generator was realized, which generates graphs with an increasing amount of nodes and edges. For performance evaluation, the complete and optimal search algorithm A* has been implemented. The quality of service is measured as the ratio of optimal divided by computed length. The simulation follows two phases: In the first phase, the problem solving agents, i.e., the agents implementing the search algorithms, are certified. In the second phase, there are two agents requesting service from the problem solving agents. The first agents selects the problem solving agent by random; the second chose that agent with the highest quality of service.



(a) box plot (b) progress

Figure 3.4: Experimental results for the quality of service

These agents generate problems, asking for the shortest path in a random graph - with a varying amount of nodes (from 5 to 500). Each of the agents creates 2.500 problem graphs. The results significantly show a superior behavior of the consumer using the certified service (cf. Figure 3.4). Further results are discussed in [Scholz et al., 2005c]. Summarizing the results, there is evidence for the benefit of certification management in this experimental setting. The prototype and experiments can be considered as a feasibility proof of our approach in this specific domain. However, this is a first result and further experiments, especially in different domains, are required for further conclusions.

Chapter 4

Engineering Autonomous Systems

The approach of autonomous software systems as well as systems of autonomous systems is applied with the perspective, that decisions usually made at design time are transferred into runtime. Doing so, the design especially with respect to requirements analysis and interaction design but also verification resp. testing is changing significantly. Here, it is possible, that contradicting requirements lead to conflicting sets of desires or performance measures within the intelligent agents. Thus intelligent agents have to be enabled to cope with such situations, e.g., by strategic autonomy [Timm and Dembski, 2005].

Engineering of autonomous software systems uses the well known features of conventional software engineering including established methodologies and tool support. Each software engineering process may be structured by the following tasks: requirements analysis, specification resp. design, implementation, and testing. Additionally, knowledge engineering captures the necessary aspects of integrating knowledge.

However there is still a gap between software and knowledge engineering methodologies as discussed in [Timm et al., 2006c]. Especially aspects in the field of validation, verification, and testing remain open. In the research presented in [Timm et al., 2006c] as well as earlier approaches in [Timm et al., 2001a] and [Tönshoff et al., 2000b] we analyzed the specific requirements of multiagent engineering and consequences for the design of intelligent agents.

In [Timm et al., 2006c] we identified mandatory process steps for agent-oriented software engineering, splitting up the conventional process steps mentioned above: requirements analysis, architectural and interaction design, semantics and dependability specification, implementation and testing (cf. Figure 4.1). In concrete software development, there might be a variation of the ordering of these steps, especially interaction and architectural design should be intertwined. As in software engineering, there are different approaches to organize the development phase: sequential, agile, and iterative processes.

Nevertheless, special requirement arise for the engineering process of multiagent

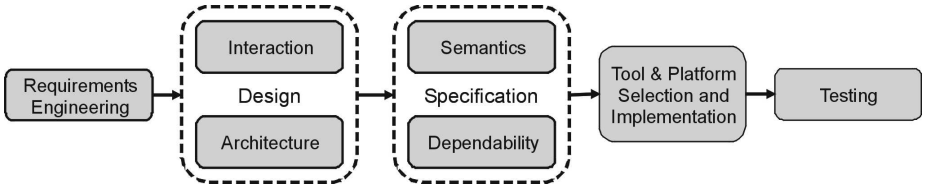


Figure 4.1: Mandatory process steps in AOSE

systems, such that conventional approaches without incorporated knowledge engineering are not sufficient. Knowledge engineering per se is not sufficient too with respect to tool or method support. In principle, engineering of autonomous systems could benefit from both: conventional tool and methodology support and the advantageous know-how about model building, knowledge representation, and inference in knowledge engineering (cf. [Schreiber and Wielinga, 1996]). Combining both aspects the development of multiagent systems requires a general theory of engineering processes.

In our work on a generalized engineering process, we focussed on three of the major challenges [Timm et al., 2006c]. In the next sections, interaction design, specification of semantics, and verification of autonomous systems is discussed.

4.1 Interaction Design

In section 2.1, we pointed out that interaction is one of the key properties of autonomous systems like agents. Interaction is used for coordination of a system of autonomous subsystems, e.g., multiagent systems. The challenge in engineering autonomous systems with respect to interaction is to create interaction designs flexible enough for emergent behavior and efficient performance but stable enough to enable reliable system behavior. The outlined interaction design presented here is based on our research on emergence and interaction in multiagent systems. For more details please refer to [Timm et al., 2001d], [Timm et al., 2002], [Timm, 2004a], and [Krempels et al., 2006].

While conventional software engineering is focused on standardized interaction and interface design engineering of autonomous systems requires flexible interaction design. In a system, autonomous subsystems have to solve problems either cooperatively or competitively depending on the application domain. [Rosenschein and Zlotkin, 1994] analyzed different application domains with respect to the interaction mechanisms by means of consideration of the world states within multiagent systems. They identified three categories ranging from general to specific: worth-oriented, where agents evaluate any state with a private value, task-oriented, where agents assess only the final states, and state-oriented, where evaluation takes place. Interaction design for autonomous systems can be supported by this classification. These application domains have strong implications to the appropriateness of specific interaction mechanisms, for example, in task-oriented domains interaction should mainly rely on behavior whereas monetary assessments as in auction protocols seem less suitable. Further details are discussed in [Krempels et al., 2006].

There are two principal approaches to multiagent communication, shared memory and message passing. Shared memory has been introduced by [Newell, 1962]. He invented the blackboard metaphor, which may be described as follows:

> Metaphorically we can think of a set of workers, all looking at the same blackboard: each is able to read everything that is on it, and to judge when he has something worthwhile to add to it. This conception is just that of Selfridges Pandemoneums' a set of demons, each independently looking at the total situation and shrieking in proportion to what they see that fits their nature.

Blackboard architectures have been widely used in the area of distributed problem solving. An advantage as well as a restriction of this approach is the fact that agents do not have to know from the existence or the address of other agents. The main drawback is that all agents have to use the same language and representation of the problem. The blackboard method has to face common problems of concurrent manipulations and the priority design.

Being aware that communication is the intentional exchange of information the message exchange approach seems to be more flexible and adequate [Russell and Norvig, 1994]. It is based on the production and perception of signs with mutual unique semantics for each communicative pair or channel. These complex structured systems of signs are arranged in so called agent communication languages [Pitt and Mamdani, 1999].

A general and conceptual description of communication via message passing (agent communication protocol) has to integrate mutual knowledge about its topics (domain), the general process of the dialog, and its current state [Carron et al., 1999]. Thus, the structure of communication protocols may be divided into a domain dependent (content, problem, topic) and a domain independent part consisting of process knowledge (methods of communication, address of partners, dialog structure). A more general approach can be found in the definition of pattern-based communication as introduced in [Krempels et al., 2006].

Standardized models like FIPA/ACL specify messages between agents by means of action, the so called communicative acts. They are defined by syntax (message model) and semantics (formal model) [FIPA CAL, 2001]. The semantic perspective is used for classification of equivalent acts. The semantics of the message is specified in analogy to AI planning by pre-conditions as indicators if an action is applicable, and by an effect describing world-state changes after the execution of the action. Rational and "irrational" behavior is captured by appropriate semantics.

The engineering process has to ensure that the agents are capable to select specific actions resp. performatives for achievement of a goal, e.g., to produce for another company. There is a strong relation to AI-planning as message sequences (conversations) are instantiations of plans. FIPA standardizes a library of reusable interaction patterns resp. protocols. As discussed in [Krempels et al., 2006], interaction protocols reduce the complexity of the plan space as the number of possible actions is limited. The FIPA-Request protocol, for example, uses 6 out of 22 possible performatives with a plan length of three actions. In consequence the combinatorial complexity is reduced

by a factor of almost 50 (63 vs. 223). This reduction ensures efficient integration of possible answers of an agent's opponent in the communication process without the need for complex plan recognition algorithms.

In the last decade, benevolence of the agents was a common assumption in multiagent engineering, i.e., the agents share common goals – at least implicitly – and there is no potential for conflicts of interest between the agents [Wooldridge, 2002]. In real-world business scenarios the assumption of benevolent agents or cooperative settings does not hold in general. Frequently, the system has to incorporate self-interested agents which do not necessarily share common goals. Theoretically, encounters of such agents resemble games where agents have to act strategically in order to achieve their goals cooperatively or competitively.

In competitive settings, the engineering of interaction has to ensure reliable system behavior even in the context of divergent agents interests. Generally, there are various mechanisms for interaction, e.g., auction protocols, bargaining, social choice. For a concrete design, the software engineer has to decide about a suitable set of interaction mechanisms. Widely used criteria for such decisions have been introduced by [Rosenschein and Zlotkin, 1994] and [Sandholm, 1999]. [Rosenschein and Zlotkin, 1994] based the decision process on the following properties: efficiency, stability, symmetry, simplicity, and distribution. They discussed these criteria in the context of the categorization of application domains as introduced before. [Sandholm, 1999] has proposed similar criteria using social welfare, Pareto efficiency, individual rationality, stability, computational efficiency, as well as distribution and communication efficiency. Obviously, both approaches are closely related and we proposed to integrate them for practical use in multiagent engineering [Krempels et al., 2006]. This unified approach is restricted to three major categories: efficiency, reliability, and complexity. Efficiency is dedicated to the assessment of individual, group, and, system outcome resp. payoff. The reliability integrates the stability criterion, e.g., defined by Nash-equilibrium, and symmetry ensuring that both sides of an interaction reach comparable payoffs. Complexity regards to technical efficiency defined by computation, communication, and speed up through distribution.

An essential requirement for successful communication remains: The interacting autonomous systems have to understand the content of the message, i.e., sender and receiver are able to decode messages with the same semantic conclusions. This does not imply, that a single specific content language has to be implemented, but mandatory requirements for content languages have to be defined. Therefore, FIPA/ACL consists of three main specifications: Communicative acts, interaction protocols, and content languages [FIPA SL, 2002], [FIPA KIF, 2003], [FIPA CCL, 2001], [FIPA RDF, 2001]. To capture the agent context, i.e., beliefs, abilities, and wants, an effective content language is often based on (multi-)modal logics in DAI. However, ontology languages like OWL can easily be adapted to the requirements and additionally offer a well structured representation of knowledge disregarding the semantics of the agent's context. Further aspects regarding the semantics are discussed in [Scholz et al., 2006]. There is an ongoing challenge in designing interaction in heterogeneous environments. Here, the agents are in need of explicit methodologies to handle heterogeneous ontologies. For further details on this problems as well as solutions refer to [Stuckenschmidt and Timm, 2002a], [Stuckenschmidt and Timm, 2002b]

and [Visser et al., 2002].

4.2 Semantics

Software engineering of intelligent agents and multiagent systems is not only dedicated to interaction design but has to deal with explicit semantics of agent architecture and interaction behavior as well as formal specification of knowledge. A short outline on semantics for representation of intelligent agents and multiagent systems is given in this section. More details are provided in [Scholz et al., 2006], [Timm, 2004b], and [Timm et al., 2002].

The formalization of agent behavior by semantics provides a descriptive level that abstracts from the agent's architecture as well as from algorithmic details. A well known example for such an abstract description is the modal logic [Wooldridge, 2000] proposed for modeling BDI agents. Formal semantics simplifies in many cases the proof that the system's specification matches the expectations about its functionality (i.e. the agent's behavior). Rao, for instance, describes a simple BDI-logic that is decidable in linear time. Such proofs perform a useful tool for the engineer as well as for the user who interacts with the autonomous agent, too. According to [Singh et al., 1999], the challenge addressed by formal semantics lies in developing "techniques for ensuring that agents will behave as we expect them to - or at least, will not behave in ways that are unacceptable or undesirable".

Engineering autonomous software systems has to keep a balance between the expressiveness of the formalism and its computational complexity, however. While even highly sophisticated formalisms may be helpful for the designer to ensure the required behavior, the internalization of such formalism into the single agents may produce problems. Consequently [Singh et al., 1999] distinguished two different objectives which can be followed by the developer of an agent formalism.

- Specification: External Use of the Formalism. This is the objective of the agent designer who uses a logical language to specify the agent's behavior. Tools from logic (e.g. model checking) are applied to analyze whether the specification is consistent.
- Reasoning: Internal Use of the Formalism: The agent's deliberation processes are implemented by reasoning within the formalism. This is the objective followed by most work on multiagent systems.

The time scale underlying both aspects of formalization is crucial. Time available for specification computations is by orders of magnitude superior to the time available for the agent's internal reasoning. Thus the price for internal use of complex formalisms may be prohibitively high though they may be helpful for engineering and analysis purpose. Agents reasoning in such formalisms would often fail to act rapidly enough under the temporal constraints of their environment. Every perception-action-cycle of a soccer agent playing in the RoboCup Simulation League for example lasts less than a second, whereas several days of computation are a common time schedule for design and model checking in order to verify the consistency of specification in a

safety-critical application. Summing up, it seems unrealistic to expect that a single formalism should be equally adapted to both, agent specification and internal reasoning.

The interaction with human users causes another type of problems. The typical user is no expert in agent technology. He tends to rely on general-purpose reasoning strategies adapted to everyday reasoning problems in a heuristic way. Such mental reasoning strategies differ substantially from algorithms employed for model checking or theorem proving. In order to optimize the user-agent interface we admit the following objective that the designer of an agent formalism could aim to achieve:

- Interaction: mental Use of the formalism. For planning his interaction with an agent, the user needs some abstract description of the agent's behaviors. It should be easy to obtain behavioral predictions by mentally reasoning with the formalism. Also, it should be possible to base high-level interaction with the agent, e.g. communication about goals on the formalism.

Although interface agents are a topic of intensive research, the suitability of agent formalisms for mental reasoning strategies has received only little attention until now.

Various formalizations defining and specifying multiagent systems and their properties have been proposed. Their majority focused on intelligent behavior within agents mostly based on explicit, cognitive models of beliefs, desires, and intentions (BDI). The underlying idea is that by observations and its actions an agent is creates an explicit world model (beliefs). Additional sets of persistent objectives (desires) and of goals which are currently pursued (intentions) creates the framework of its behavior. The agent pursues its goals by autonomously created plans. BDI-agents are "the dominant force" in formal approaches [d'Inverno et al., 2004] for which [Wooldridge, 2000, p. 7] already identified three major reasons:

- It is based on a widely accepted theory of rational actions of humans
- They are successful in a great number of complex applications
- There is a large family of well-understood, sophisticated, and formalized approaches available

Standard BDI approaches however do not focus on system behavior but on agent internal knowledge representation and decision making. [Wooldridge and Lomuscio, 2000] introduced VSK as a formal model for the entire multiagent systems based on (multi-)modal logic. This approach takes into account that different agents create different pictures of the world, which may be only partly visible in general and especially for the single agent. Formally a (global) visibility function (visibility) and an agent depending perception function (see) are introduced in addition to local states (knowledge) in terms of (multi-)modal sorted first order logic including the possible worlds semantics of BDI. While the underlying idea of VSK is convincing, it suffers from the problems introduced before. For further details on formalization of agents, agent's behavior, and multiagent systems refer to [Scholz et al., 2006], [Timm et al., 2002], and [Timm, 2004b].

4.3 Testing

An essential part of software engineering is dedicated to verification and validation. Non-determinism however makes verification of AI software difficult, this difficulty is increased if concurrent, distributed or object-oriented systems as multiagent systems are in question. Evaluation and validation became bottlenecks for agent technology as there is no standard for verification and assessing available ensuring adequate quality of multiagent systems, i.e. the fulfillment of predefined requirements. To overcome this gap of a glance towards mainstream software quality approaches may be useful. In conventional software engineering the software product quality can be defined as follows [Riedemann, 1997]:

- Functionality is a set of properties with respect to the existence of a set of functions that implement the specified requirements:
  - Adequacy indicates the existence or applicability of the software for the specified tasks,
  - Correctness is used for deciding if the results or effects of a software are correct,
  - Interoperability considers the applicability to interact with predefined systems,
  - Normative adequacy is used for estimation of satisfaction of application specific norms or commitments or juridical rules of the software,
  - Security deals with aspects of unauthorized access to the program or data.
- Dependability is the capability of the software to keep a performance portfolio in predefined conditions over a specified time period and is defined by the properties of:
  - Maturity is used for determining the frequency of failures or fault states,
  - Fault tolerance describes the appropriateness with respect to a predefined performance level of a software in situations where either unspecified access to interfaces or soft-ware failures occur,
  - Recovery is the possibility of a software to be recovered on a prior performance level including retrieval of data in adequate time.
- Usability is the property which is related to the effort required for using the software as well as an individual assessment of using the software by a predefined group of users.
- Efficiency is a set of properties which indicates the ratio between the performance level of software and the amount of used resources in predefined conditions.
- Adaptability is related to the necessary effort for performing given modifications (corrections, improvements, or adaptations)

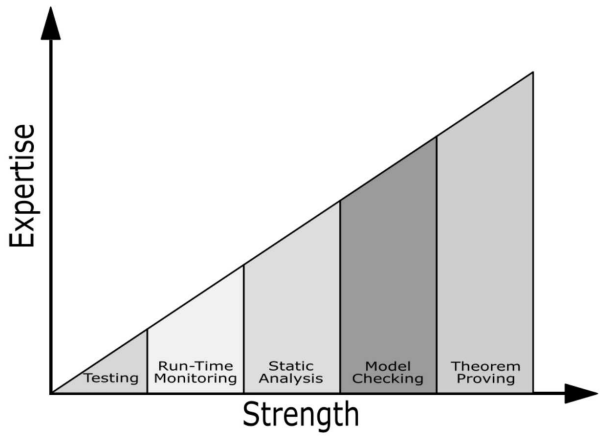


Figure 4.2: Classes of evaluation in AI

Quality management covers constructive as well as analytic activities [Riedemann, 1997]. Constructive quality management ensures distinct properties of the emergent product during analysis, design, and implementation phases of the engineering process. Analytic quality management analyzes the fulfillment level of the quality properties by verification, validation and evaluation. As [Hoppe and Mesegeur, 1993] emphasize that misconceptions arise if terms like verification are used in a different way as in artificial intelligence we apply the term evaluation to express the determination of how far agreed, prescribed, or expected features of an object are fulfilled. [Menzies and Pecheur, 2004] differentiate evaluation with respect to the required amount of expertise and strength of proof: testing, runtime monitoring, static analysis, model checking, and theorem proving (cf. Figure 4.2).

Following this classification software systems may be evaluated by formal proofs of correctness (theorem proving) with highest strength of proof and highest need for expertise or with less effort of expertise and less strength of proof by model checking of formal specification and model satisfaction. In contrast to static analysis, runtime monitoring analyzes systems by their runtime behavior with specified input parameters resp. predefined conditions. Testing uses special programs that simulate input sequences and analyze the results with respect to the requirements. Here the expertise (insight into the black box) may be short but the strength of proof is poor, too.

Formal approaches as discussed in the last section are widely used for agent design. Formal representations of requirements became a key tool for developers or customers in order to validate or even verify that the software system meets the requirements. However, the correctness proofs are at most not terminating or decidable for formal languages like multi-modal logics or first-order logic. Thus formal verification is often limited to very specific aspects. In current applications, description logics are widely used for representing communication content such that formal prove of interaction behavior should be feasible.

The evaluation of intelligent agents and agent systems though far more complex may profit from the conventional software evaluation approaches outlined so far. The first expansion is that reflecting the autonomy of the agents requirements have to be specified primarily as goals or context for agent behavior. Furthermore, specification of formal requirements has to include complete state models of the environment and the agent. Common agent-oriented methodologies contain a software engineering process, which in some cases like MaSE [DeLoach, 2004] or Prometheus [Padgham and Winikoff, 2004] is closely related to object orientation. While early object oriented methodologies claimed for a natural reduction of evaluation efforts and enhanced quality later studies showed that the actual error rate in object oriented code is even higher than in conventional software. The increased modularization is accompanied by increased but mainly implicit interdependencies between modules. Cross reference of methods or classes, hidden information (esp. polymorphic inheritance) and high dimensional possible states are further sources of problems.

One has to be aware that the process of evaluation of multi agent systems will suffer from similar difficulties, but another dimension of complexity evolves from the key feature of multi-agent systems, emergent effects, emergent properties or emergent organization. Emergence, often considered as non-deterministic, arises at runtime. It produces flexible solutions (emergent effects), constellation and types of agents. The system properties may be changed (emergent property) and new interaction schemes may occur (emergent organization). Emergence is per definition not specified in advance leaving the system partially unspecified. The evaluation of a system that contains properties that are not specified and modeled in advance results in an incomplete analysis.

Emergence is not the only source of obstacles for the evaluation process of multi agent systems: Agents are assumed to act autonomously following there individual goals. Obviously, this may cause concurrency problems. Evaluation of concurrent software is an enormous challenge even in conventional software engineering. There is still another complication compared with conventional software evaluation: non-deterministic environments may cause additional problems increasing if mobile agents are in question.

Many open problems around these aspects have been identified in the agent community, but there is still a comparably small amount of publications which propose convincing solutions. In concurrent system evaluation for example, [Riedemann, 1997] proposes explicit control on schedule, time, and invocation of methods. [Gomez-Sanz et al., 2004] reported that current research is focused on formal verification. Formal approaches however -if applicable in the special case- often suffer from complexity requesting unrealistic high expertise from designer and user and overextended computational efforts, which is not feasible for most practical purposes in the engineering process. Few methodologies of multiagent engineering incorporate evaluation tools in an adequate way, e.g., Tropos [Mylopoulos et al., 2001].

The state of the art surveys from [Weiss and Jakob, 2005] however emphasize the necessity of verification in real-world applications.

Summing up, the evaluation intelligent agents and multiagent system is a challenging task. We propose to develop solutions with respect to any of the classes of evaluation (testing, runtime monitoring, static analysis, model checking, and the-

orem proving) depending on the field of application. In the following, we discuss approaches to testing and runtime monitoring. For further approaches please refer to [Timm et al., 2006a].

Testing

In conventional software engineering there are multiple testing methods available which may be applied to the agent engineering process. A prominent approach is the unit test during the implementation of functions, methods, interfaces, or classes. Its key benefit is its repeated and automatic application. The tests are automatically performed after each change in the unit, even if the changes are not directly connected to the test object. Unit tests working on object oriented implementations are well supported by many tools and development environments. We propose to apply the unit test methodology to the design and implementation of agents, since their structural implementation is usually object oriented. In order to adapt the unit test to multi agent system engineering one has to address specific problems as described above. Knowledge representation, inference as well as interaction resp. communication in the multiagent system have to be considered.

Concerning knowledge representation conventional unit test approaches are applicable as long as the representation is implicit because the knowledge representation is part of the code and architecture in this case. In more sophisticated architectures, explicit knowledge mainly based on description logics (ontologies), or Prolog is used. Such representations are suitable for model checking or theorem proving, which can be combined with conventional unit testing, for example to execute a description logic reasoner within object-oriented code.

Interaction in multi agent systems is emerging at runtime causing special challenges for the testing procedure. There are boundaries for this emergence however, usually specified by valid sequences of messages, i.e. interaction protocols with mandatory interactions and propositions on the content. Tests should check the agents' compliance with these interaction boundaries. The special unit test proposed here considers the agents as units and constructs a controlled test environment for them. For definition of allowed sequences we propose to use grammar as syntactical specification. Messages in an interaction are considered as words of a language and sequences as sentences. Structural consistency is tested by application of syntactic checkers like parsers. The content of messages may be tested matching content on equivalence, e.g., string comparison or by use of matching algorithms for formal content representation, e.g., subsumption for description logics. Especially such formal content representations constitute an effective compromise between computational complexity and problem adequacy.

Testing nondeterministic behavior however, makes the test far more complex than for conventional object oriented software. It has to specify the environment with its variation and analyze the variance of the expected results resp. sequences of the interaction.

An interaction test framework that incorporates these aspects is shown in figure 4.3. The test environment consists mainly of the tester with test case generator for the generation and selection of appropriate test cases. These are assigned to concrete mes-

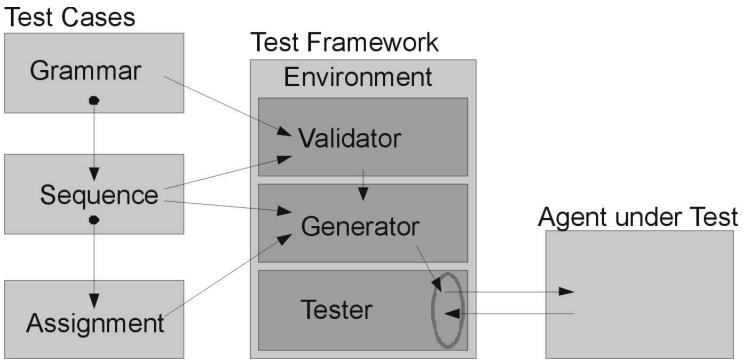


Figure 4.3: Interaction test framework

sages which are evaluated by the validator component using the specified grammar and the set of allowed sequences. This framework facilitates the test of agent interaction behavior in a controlled way without postponement of the test to the runtime of the system. In order to establish this interaction test procedure in practical engineering of multi agent systems, however, tool-support is needed.

Runtime Monitoring

Runtime monitoring establishes a controlled environment and a control structure for tests under specific configurable conditions, too. The focus of this method of evaluation is the system behavior and not the specific units, however. Thus the distributed and partially non-deterministic character of the system are more in the center of runtime monitoring compared with unit tests. The evaluation by runtime monitoring has to deal with a situation where internal structures or details of coding are hidden or the non-deterministic behavior of agent inferences causes problems, prohibiting simple white box settings. Thus grey- or black-box approaches have to be used. In our paper [Timm et al., 2006a] we discuss two pragmatic approaches to runtime monitoring: simulation covering both agent and multiagent runtime monitoring, and certification management only for agent runtime monitoring. In the following we restrict the discussion to some general aspects of the first aspect: simulation for runtime monitoring.

Even if there is no domain-independent tool support available for these approaches, the methodologies can be transferred to concrete engineering projects.

This method uses a controlled environment as well as a control structure and adds a specific agent collecting and documenting information about the dynamic system behavior. Moreover a random generator is incorporated establishing a stochastic simulator in order to cope with great variations and unforeseeable states of environment and multi agent system. The stochastic simulator triggers the control unit and establishes varying conditions in a high number of sequential runs. By this, the test procedure becomes a stochastic process and statistical analysis of the system behavior becomes possible. In this setting (considered as a grey-box test) it is possible to evaluate the dynamic and adaptive behavior of the agents and the system. The length of each simu-

lated run of the system depends on the dynamic characteristics and requirements to be tested. If the system is supposed to reach a balanced state with respect to the relevant parameters within a given time t0 for example, this requirement will cause a length of at least t0 for the length of each simulation run. Other relevant parameters of the test procedure may be derived from the set of agentsgoals resp. system requirements using analogous arguments. Statistical analysis is focused on both aspects, the dynamical behavior looking at the parameter development over time (time series analysis) and the summarizing results (reached state at the end of the run or mean values). Special statistical methods fit to these tasks, for example t-test, Kruskal-Wallis-test, ANOVA (analysis of variance). [Timm, 2004b] developed a structured testing environment for runtime analysis of multi agent systems and proved its usability in different application scenarios and domains (cf. Chapter 2.4 and 2.6, [Timm, 2004b]).

Chapter 5

Applications of Autonomous Software Systems

In the last chapters, autonomous software systems have been introduced with respect to levels of autonomy, strategic management, and engineering issues. In the following, we will present our research on applications of autonomous software systems, i.e., intelligent agents. The three applications presented here, are dedicated to the general domain of logistics covering the different scenarios and research issues of process planning and production control, mass customization, and autonomous logistic processes. Starting with IntaPS, we introduce a multiagent approach for integration of process planning and production control. Even though the requirements for the internal reasoning within an agent are high, the interaction complexity is limited. In the next section, we discuss the impacts of high degree of individualization in manufacturing and logistics (mass customization) to the design and implementation of business information systems. The logistic chain of mass customization is optimized by the application of intelligent agents. The third domain, takes the application of autonomous systems to the next level: agents are widely used in order to enable self-organization in logistics.

As introduced in the section 4.3 on testing of multiagent systems, it is essential to test and evaluate agent applications. In the context of shop floor systems resp. process planning and production control, we have applied simulation successfully for the evaluation. However, if we consider the evaluation of self-organizing logistics in realistic scenarios, computational complexity as well as memory usage may overextend standard hardware platforms. Thus, we discuss the application of grid-technologies for dynamic resource management in large scale multiagent simulation in logistics.

5.1 Process Planning and Production Control

Conventional approaches to manufacturing logistics focus on the economical scale of the production (enterprise resource planning), i.e., production planning and control systems, are dedicated to the technological information of the product disregarding concrete production planning and control (product data management).

Several applications of agent technology in manufacturing focus on digital marketplaces where intelligent agents act on behalf of enterprises, customers or other organizations to achieve goals like ”acquire a specific good for the smallest possible price”. Since enterprises began to consider several departments as individual profit centers, market-based coordination mechanisms became of increasing concern not only for inter-enterprise relationship but also for internal processes of enterprises like e.g. scheduling tasks at the shop floor. These agent systems are autonomous on an operational level with respect to the classification introduced in Chapter 2. Considering customized products in small lot sizes produced as a batch job, scheduling becomes a distributed problem: Orders have to be manufactured by different resources located at different places at the shop floor. Each resource possesses its own schedule, its own capabilities to perform different manufacturing operations and its own economical profile (e.g. specific machining costs). On the other hand, orders need to be manufactured according to customer requirements and due dates. From point of view of the shop floor, an order schedule must be ”calculated” in cooperation of order and resources, where each individual resource decides about the price it will offer its capabilities to the order. Unfortunately, the structure of the shop floor and the orders are not a static since new machines are taken into operation, other suffer a breakdown, or typical orders change due to altered customer demands: The shop floor for customized manufacturing is a very dynamic environment. Following the argumentation in Chapter 2, agent technologies can be used to overcome traditional problems incorporating strategic or tactical levels of autonomy.

The conventional management of manufacturing is based on a tayloristic separation of preparatory activities in planning and implementation activities [Toenshoff, 1999]. Planning activities range from specification of the product, e.g., workpiece geometry, of the process, e.g., required production methods, to selection of resource types, e.g., machine tool type. These aspects are more connected to the technological information and implementation activities are dedicated to economical information, i.e., amount of products to produce, e.g., specification of lot sizes, planning and scheduling of the process steps, selection of resources for production, as well as planning of the human resources. This approach results in a gap between the involved systems, such that a dynamic interaction is not possible, i.e., potential loss of time and information occurs. This gap may be bridged by autonomous systems, which integrate the planning and implementation systems. However, the operational autonomy is not sufficient for this problem, as relation between resource capabilities and product features are hard-wired and cannot be modified at runtime. This requires tactical autonomy for reallocation of resource-consumer-links. Thus, the IntaPS approach implements a system of intelligent agents, which are capable of integrated process planning and production control in a very flexible and distributed way. The basic architecture of the IntaPS approach is illustrated in figure 5.1.

In IntaPS, intelligent agents integrate the two substantial components of earlier stages of product development and the resources on the shop floor. This link is realized by decentralized planning on shop floor level, i.e., electronic marketplace, and by rough level process planning. The system consists of three different types of agents: resource agents, order agents, and service agents enabling decentralized planning on shop-floor level. The resource agents are representing relevant resources of the phys-

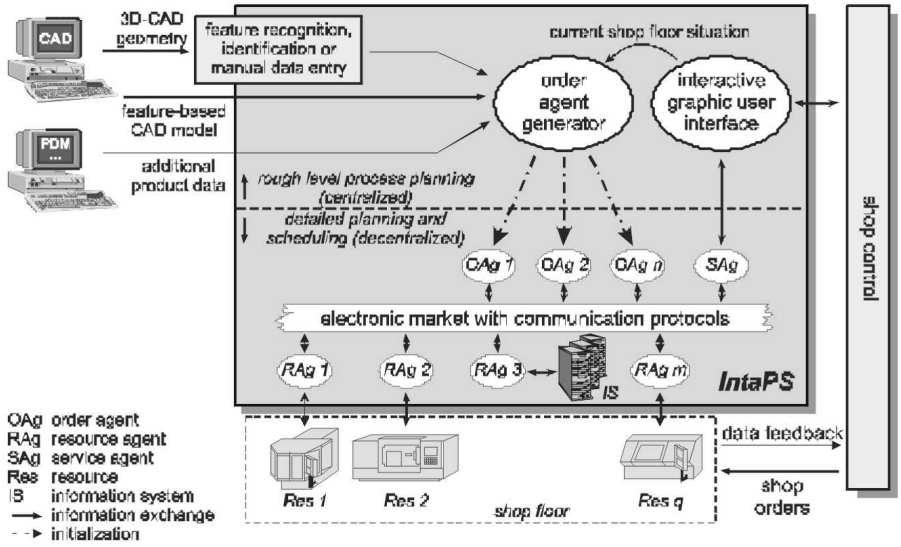


Figure 5.1: The IntaPS architecture

ical production system, like machine tools, transportation devices, human resources, or even virtual resources, e.g., information systems together with its real-world environment. These agents incorporate the relevant information with respect to process planning and production control as part of their knowledge bases, e.g., capabilities of the machine tool or product features. Order agents represent product orders, which have to be manufactured. Due to its autonomy and pro-activity, an order agent is able to recognize internal and external disturbances and to react appropriately. The third type of agents (service agents) is dedicated to maintenance and observation of the system. Furthermore, service agents can be used for implementing strategic management by reflection as introduced in Section 3.3 or for evaluation and verification purposes, e.g., testing and simulation management.

An electronic marketplace is used for the dynamic coordination of order and resource agents. Next to the tactical autonomy, applied here for the match-making of resource capability and product feature, operational autonomy improves coordination in the shop floor dynamically. Order agents and resource agents interact according to three mandatory stages: negotiation, verification, re-negotiation.

Starting with a negotiation, orders communicate required manufacturing skills or product features and due dates. Thus, order agents identify appropriate resources, which are capable to perform a specified product feature resp. manufacturing task with respect to logistical constraints. These logistical constraints take organizational constraints into account as well as limitations in time, e.g., due dates specified by the customer. The negotiation on product feature and capabilities follows the ontological manufacturing knowledge which is determined by standardization like ”STEP-NC” ISO 14649. Thereafter, suitable sequences of manufacturing operations result from

auctions between identified appropriate partners. The optimal sequence of manufacturing operations is accepted as detailed plan. Thus, some of the traditional tasks of process planning are carried out in a distributed manner.

The second stage is dedicated to the verification of the detailed plans. The order agent examines continuously whether its detailed plan is executable under the current conditions of the shop floor and its resources. Disturbances (e.g. breakdown of a machine tool) or emerging chances (extended machine capabilities) cause order agents to analyze the consequences and to identify those parts of their detailed plans, which are affected. In this case, the order agent decides whether a modification of the plan is required.

The order agent initiates the re-negotiation stage if necessary and tenders parts of the detailed plan for a new negotiation. The re-negotiation phase should lead to an improved alternative detailed plan which substitutes the previous plan. Afterwards the verification phase is resumed and lasts until the order is finished.

While the order agents act mainly on an operational or tactical level of autonomy, the resource agents are more sophisticated within IntaPS. Additional and potentially conflictive requirements arise from various objectives, which have to be considered by the resource agents. A classical example is the conflict between the goals of ”short lead time” and ”small stock size”. Hence, the cobac algorithm to enable strategic autonomy in the resource agents is integrated (cf. Section 2.6).

In addition to decentralized components of the multiagent system, the IntaPS project specifies a centralized rough level planning, which integrates preprocessing of incoming data and generation of rough process plan. The underlying data are given by geometrical or technological information about the product, further organizational information related to products and orders. Order agents are synthesized on the basis of this information and initialized with a rough level process plan. Furthermore, the centralized part of the IntaPS architecture provides a graphical user interface for interaction with the system. Thereby, IntaPS serves as an example for the application of different levels of autonomy for the integration of two different information systems within an enterprise (process planning and production control).

The IntaPS project is funded by the Deutsche Forschungsgemeinschaft from 2000 until 2006. Various articles have been published, containing more details on the approach outlined here, e.g., [Lorenzen et al., 2006], [Timm et al., 2001c], [Tönshoff et al., 2001b], [Tönshoff et al., 2000a], [Tönshoff et al., 2002a], [Tönshoff et al., 2002b], and [Tönshoff et al., 2002c].

5.2 Logistics in Mass Customization

Consumers tend to have precise ideas for their product demands on a very individual level. The management concept of mass customization is an answer to this tendency of consumer driven production [Piller, 2003]. By combining individuality of customized goods with adequate low prices as known from mass production, this concept has been considered to reach high customer’s satisfaction. We assume that future business information systems will use service-based computing, i.e., web services and networking infrastructures like the internet, for collaborative business. The joint research project

EwoMacs[1], analyzed logistics structures of mass customization in the shoe industry. The research performed at Technische Universität Ilmenau was focused on the engineering aspects, i.e., the question of how to design a multiagent system on the basis of business processes. Thus, we introduce the results of this project as illustration for the engineering of real-world applications.

The project aims to build a system coordinating the respective information logistics. Mass customization is a challenge for the design and implementation of business information systems of the participating enterprises as a great demand of information has to be distributed, interpreted, and processed [Pawlaszczyk et al., 2004]. In order to reduce transaction costs a high degree of automation for information logistics is required. Moreover the supply chain itself seems simply to design an efficient information system (cf. Figure 5.2). However, complexity of the information handled in this process overextends the capabilities or the costs of conventional centralized data exchange processes [Pawlaszczyk et al., 2004].Insufficient coordination between actors, adversely exploited autonomy of each actor and failing in information transmission are severe problems for smooth configuration, production, and delivery of MC products. Hence a system of decentralized and autonomous acting intelligent agents has been used to overcome these problems. The complexity of the contents of transferred information and the different understanding of short communications in everyday language indicates the use of explicit knowledge representation in this system. Ontologies have been considered as a promising approach in this situation. The ontology has to ensure that the customer's requirements have to be integrated in a product specification and to supply each actor of the value chain with information about the product and customer.

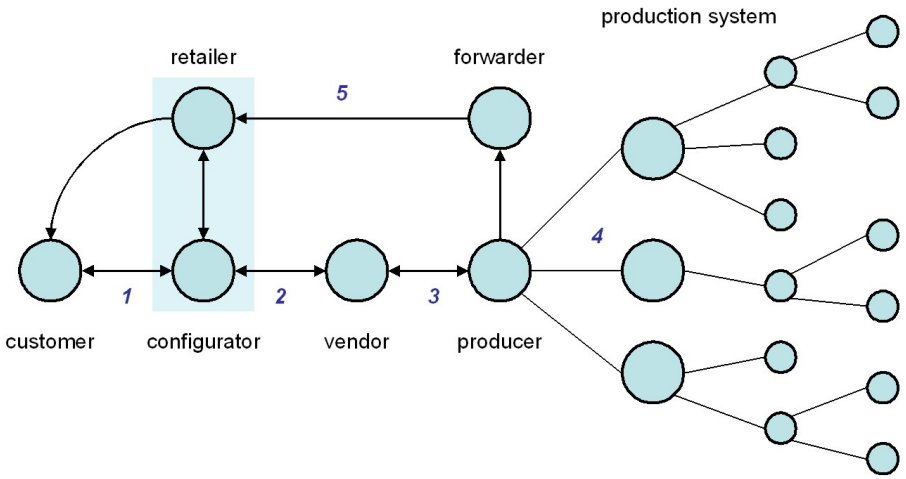


Figure 5.2: Actors in mass customization

The EwoMacs system deals with the integration of different types of systems, e.g. production planning, process control, enterprise resource planning, involved into the

[1]EwoMacs was funded by the German Ministry for Education and Research (BMBF) from 2002 to 2004.

process of customizing, producing, and delivering. There is need for integration of the involved information systems - at least virtually. Here, problems of data privacy and security arise as more than two independent companies are interconnected. Only uncritical data needed for the common processes should be exchanged. The design process had to overcome the problem of missing or inadequate standardizations for data exchange and various, partially contradictory definitions of used concepts. To address this problem, our approach is based on agents as autonomous software systems, which are representing enterprises for the automated cooperation in logistic networks. The multiagent system provides a framework for cooperation within short-term relationships [Knirsch and Timm, 1999].

The conventional approach to design interaction relationships in economy is to perform a business process analysis. Modern tools for business process modeling like ARIS [Scheer, 1995] integrate simulation functionalities for deep analysis. We propose to use this analysis as a starting point for efficient multiagent engineering in this domain. Therefore, we developed the DAISIY (Deliberative Agents for Intelligent SImulation SYstems) framework. DAISIY explicitly integrates simulation is a key aspect. Instead of developing a single system supporting mass customization in the real world, the simulation is used for analysis and identification of scaling up problems in the application, etc. Furthermore, a highly distributed system, which contains modules and autonomous systems from heterogeneous enterprises, can hardly be tested properly (cf. Section 4.3). Thus, simulation is also used to ensure reliable behavior of the system. The approaches to this domain are two-fold. On the one hand, simulation systems are used for analyzing the domain, planning and re-engineering of business processes. On the other hand, development of management and control systems for information flow and business processes is of interest.

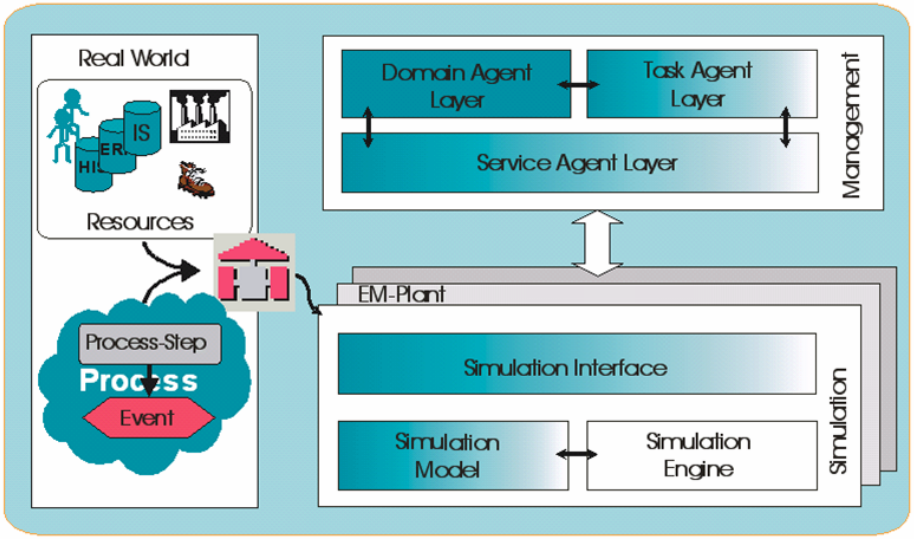


Figure 5.3: The DAISIY approach

The DAISIY framework integrates a methodology for analysis, design, simulation, and implementation. The aim of DAISIY is to guide researchers or practitioners by analyzing the domain, designing processes, and implementing an agent-based control or simulation system for a specific scenario. DAISIY introduces a five-step methodology as illustrated in Figure 5.3 [Pawlaszczyk et al., 2003]:

1. Business process modeling using a standard language and tool

2. Transfer of this model to a simulation model

3. Automatic synthesis of a generic agent system on the basis of the simulation and business process models

4. Enhancement of the agent model: intelligent agents as substitution for autonomous entities along critical paths, in order to preserve structure and behavior of the real world system

5. Detailed simulation using the multiagent system

For engineering real world applications, the extension of this approach is required. As mentioned in section 4, a standardized, iterative process should be applied, following the mandatory steps: requirements analysis, architectural and interaction design, implementation, and verification. The DAISIY approach with focus on simulation should be integrated in engineering process to meet the requirements of the real-world. Based on the layered architecture in the DAISIY framework three aspects: real world, simulation and management, are considered. These aspects are addressed to introduce respective components: modeling for the real world, multiagent system and simulation. Further details on this approach may be found at [Dietrich et al., 2003], [Dietrich et al., 2006], [Timm et al., 2001b], [Timm et al., 2001e], [Pawlaszczyk et al., 2004], [Timm and Pawlaszczyk, 2002], and [Pawlaszczyk et al., 2003].

5.3 Autonomous Logistic Processes

In the last decade, logistics has changed enormously. Various change driver lead to a situation, where complex and partially conflicting requirements on logistic planning and control systems arise. However, currently available strategies, methodologies, and tools lack limited efficiency in this context [Scholz-Reiter et al., 2004]. An emerging approach in research may be found in the analysis and design of autonomous systems in the operationalization of logistic processes. The objective of autonomy here is to enable processes to interact cooperatively for individual as well as global optimization. To meet the new requirements of modern logistics, new ways of designing, implementing, and managing logistic processes are required. Therefore, the German national science foundation (Deutsche Forschungsgemeinschaft) is funding the Collaborative Research Center on ”Autonomous Logistic Processes - A Paradigm Shift and

its Limitations" (SFB 637) at Universität Bremen[2]. Autonomous cooperation describes processes of decentralized decision-making in heterarchical structures. It presumes interacting elements in non-deterministic systems which possess the capability and possibility to render decisions independently. The objective of autonomous cooperation is the achievement of increased robustness and positive emergence of the total system due to a distributed and flexible coping with dynamics and complexity. The research in the CRC addresses the management resp. decision system, the information system, and the technical system.

The technology of autonomous systems especially multiagent system is a core research object within the CRC as it has the capabilities to integrate the different levels of a logistical system. The integration of technical level and information level seems to be obvious. RFID chips can be used as physical container for agent software and agent software can interact with the real world by sensors and actuators, for example. However, the decision making in autonomous software systems requires knowledge and consequently a knowledge management system. However, as in logistics autonomous subsystems of various enterprises interact, there is the need of competitive knowledge management. Additionally, knowledge management has to meet requirements of real-time, distribution, and high dynamics. Conventional knowledge management systems managed individually by the enterprises can hardly incorporate process knowledge on collaborative logistics processes. Thus, we propose an emergent knowledge management approach [Langer et al., 2005]. Each autonomous system can incorporate roles, which are related to knowledge management functionalities. In runtime, the cooperation of various autonomous systems interacting on the basis of these roles is creating a knowledge management system. Cooperation or competition is integrated by specific interaction patterns, e.g., in competitive environments, knowledge retrieval has to be paid in a real or virtual currency; the amount has to be negotiated by agent communication.

While logistics is often handled by the definition of logistic networks, the innovative approach of the CRC is to address the logistic process itself. Consequently, the degree of complexity of decision making should be reduced as not any consequences have to be considered with respect to the network. However, there is a potential loss of control, i.e., it has to be ensured, that the locally interacting autonomous subsystems act with respect to the whole system. To enable dependable multiagent systems, a subproject deals with the explicit integration of risk management to multiagent systems [Lorenz et al., 2005].

However, the integration of managerial aspects requires more sophisticated approaches. The strategic management introduced in this paper tries to bridge the gap between the information system, which can obviously be handled by autonomous systems, and the management level. From the management perspective, this requires the delegation of decision-making competence from the managerial to the information level. In consequence, the management is losing influence on local decisions. To compensate this loss of control new instruments and logics are needed for transparency and dependability issues [Hülsmann and Berry, 2004]. In a standard approach to lo-

[2]The research on autonomous systems introduced within this paper is closely related and partially funded by this CRC.

gistics, autonomous subsystems act on the basis of pre-defined set of rules, short-term objectives of the enterprise, and current information about their environment. This approach allows for operational autonomy on the process level. If the tactical or strategic level of autonomy should be applied, the decisions delegated to the information system are of increased complexity as aspects usually handled by the management, e.g., contradictions in objectives. The integration of management aspects in the decision making of autonomous software systems is an open research issue; further details are published in [Dembski and Timm, 2005].

In our research, applications in the logistics domain are accompanied by research on knowledge and risk management as well as contradiction management. For further details refer to [Dembski and Timm, 2005], [Langer and Timm, 2004], [Langer et al., 2005], and [Hammer et al., 2005].

To explore realistic behavior of systems of autonomous cooperating processes, it is necessary to model large scale application systems, which can be simulated and analyzed properly. Next to simulating the material flows of these nets it is important to integrate the information flows, as especially autonomous processes are depending on data, information, and knowledge within the environment. Current research projects within the CRC are studying the effects and influence of available communication networks in the context of large scale logistic networks. To analyze effects like lack of communication it is required by these projects to simulate huge amounts of autonomous systems (10^4 to 10^6 agents). As conventional multiagent simulation approaches are not capable of handling these amounts of actors, the question arise if grid technology could be a beneficial approach to realize large scale multiagent simulation.

The simulation of information intensive large-scale agent systems necessitates high computational power. Distribution of simulation models is applied for coping with these challenging requirements. However, the distribution of the simulation model on different computers leads to further problems, i.e., the synchronizing of events to ensure causality and monitoring of the distributed simulation state, load balancing, as well as dynamic resource allocation [Timm and Pawlaszczyk, 2005]. To address the issues of failure resistance, load balancing, transparent executing of large-scale simulation experiments in logistics, grid technology seems to be a helpful infrastructure for resource allocation and consequently speed up of simulation experiments. We propose a service-oriented, decentralized, grid-based architecture approach for multiagent based simulation (cf. Figure 5.4).

In this architecture, the P2P-Grid layer is responsible for dynamic resource allocation. The middleware implements required simulation functionalities, e.g., supporting a set of services to distribute passive and active simulation objects (the agents) on the network or to initialize, start and stop simulation runs. The middleware is also dedicated to event-synchronization, monitoring and reporting. The Agent runtime environments (Agent RTE) provide the infrastructure for the domain dependent autonomous systems, i.e., concrete simulation experiments are located at the top-level layer utilizing the lower level services.

We explored the feasibility of this architecture using the well known traveling salesman problem with autonomous systems, which implement evolutionary algorithm for problem solving. Doing so, the scalability of the system was tested by executing several thousands computation runs while changing the number of nodes in the Grid environ-

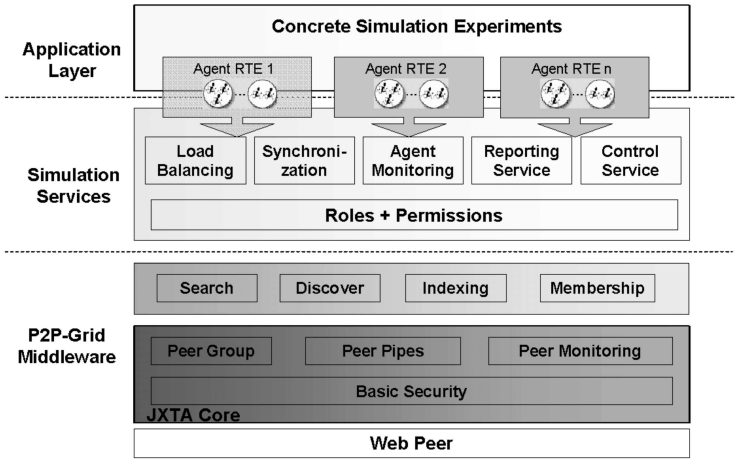


Figure 5.4: Layered architecture

ment. The results show a superior behavior, i.e., speed up is reached which is significantly higher than the additional communication costs introduced by our approach. We observed almost a linear speed up, decreasing computation time by averagely 40 percent when doubling hosts. For further details on this approach, the experiments, and the results, please, refer to [Timm and Pawlaszczyk, 2005].

Chapter 6

Conclusion

Autonomous software systems are a promising approach to cope with increasing complexity within modern application fields. Here, agents can be used to realize autonomy on different levels. The levels of autonomy introduced in Chapter 2 can support the requirements engineering step within multiagent system design to determine which agent architecture is appropriate for which problem resp. task. We introduced different approaches to integrate autonomy on the respective levels within intelligent agents. Simulation results showed the superior behavior of system incorporating operational and strategic autonomy. The adequacy of capability management enabling tactical autonomy has to be shown by prototypical implementation and simulation.

Furthermore, the research illustrated in this paper together with the results of the priority research program on "Intelligent Agents and Business Applications" (cf. [Kirn et al., 2006]) indicate, that autonomous systems, i.e., intelligent agents, are beneficial if flexibility is required. The research issue discussed in the second part of this paper is: "If the agents optimize themselves locally, does a global optimization emerges?" To solve this problem, we proposed to introduce strategic management as a runtime component in multiagent systems. But next to this question, it is an important to adjust the level of flexibility resp. autonomy needed in the domain. In consequence, unplanned events should not automatically initiate a global system change. Reflection as introduced by Luhmann can be transferred to multiagent systems for bounding flexibility in dynamic manner. This approach meets the challenge of strategic management of autonomous systems to keep as much autonomy as possible in the subsystems and integrate a methodology for adjustable autonomy in the runtime behavior. We addressed this challenge of strategic management from different perspectives (Chapter 3) with the approaches of versatile management, reflection, and certification management. These approaches constitute basic methodologies coping with high dynamics in the application domains.

The strategic management of autonomous systems includes the software engineering process, too. In this paper, we analyzed the software engineering process and identified interaction design, semantics, and evaluation resp. verification as key challenges. For each of these aspects, we proposed a systematic approach with special focus on multiagent system engineering. Especially the problem of verifying and testing of au-

tonomous software systems is in focus of ongoing research in the agent community. We introduced new approaches dealing with testing and runtime monitoring as special classes of verification.

Bibliography

[Alexander and Giesen, 1987] Alexander, J. C. and Giesen, B. (1987). From reduction to linkage: The long view of the micro-macro link. In Alexander, J., Giesen, B., and Münch, R., editors, *The Micro-Macro Link*, pages 1–45. University of California Press, Berkeley.

[Axelrod, 1997] Axelrod, R. M. (1997). *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton Studies in Complexity. Princeton University Press, Princeton, NJ.

[Boden, 1977] Boden, M. A. (1977). *Artificial Intelligence and Natural Man*. Basic Books, New York.

[Bohman, 1993] Bohman, J. (1993). *New Philosophy of Social Science - Problems of Indeterminancy*, chapter The Macro-Micro Relation, pages 146–185. The MIT Press, Cambridge.

[Carron et al., 1999] Carron, T., Proton, H., and Boissier, O. (1999). *Multi-Agent System Engineering - 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent Worlds, MAAMAW'99 in Valencia, Spain*, volume 1647 of *Lecture Notes in Artificial Intelligence*, chapter A Temporal Agent Communication Language for Dynamic Multi-Agent Systems, pages 115–127. Springer.

[Castelfranchi and Conte, 1992] Castelfranchi, C. and Conte, R. (1992). Emergent functionality among intelligent systems: Cooperation within and without minds. *Journal on Artificial Intelligence and Society*, 6(1):78–87.

[Conen and Neumann, 1998] Conen, W. and Neumann, G. (1998). *Coordination Technology for Collaborative Applications*. Number 1364 in Lecture Notes in Artificial Intelligence. Springer, Berling.

[de Wolf and Holvoet, 2005] de Wolf, T. and Holvoet, T. (2005). Towards a methodology for engineering self-organising emergent systems. In Czap, H., Unland, R., Branki, C., and Tianfield, H., editors, *Self-Organization and Autonomic Informatics (I)*, volume 135 of *Frontiers in Artificial Intelligence and Application*, pages 18–34, Amsterdam. IOS Press.

[DeLoach, 2004] DeLoach, S. A. (2004). The mase methodology. In Bergenti, F., Gleizes, M.-P., and Zambonelli, F., editors, *Methodologies and Software Engineering for Agent Systems*, volume vol of *ser*, pages 107–126, Boston. orga, Kluwer Academic Publishers. notea.

[Dembski and Timm, 2005] Dembski, N. and Timm, I. J. (2005). Contradictions between strategic management and operational decision-making - impacts of autonomous processes to decision-making in logistics. In Palwar, K. S., editor, *Innovations in Global Supply Chain Networks. Proceedings of the 10th International Symposium on Logistics (ISL 2005)*, pages 650–655, Lissabon.

[Dietrich et al., 2003] Dietrich, A. J., Timm, I. J., and Kirn, S. (2003). Implications of mass customization on business information systems. In *Proceedings of the Second World Congress on Mass Customization & Personalization*.

[Dietrich et al., 2006] Dietrich, A. J., Timm, I. J., and Kirn, S. (2006). Implications of mass customization on business information systems. *International Journal of Mass Customisation*, page zur Veröffentlichung angenommen.

[d'Inverno et al., 2004] d'Inverno, M., Luck, M., Geogreff, M., Kinny, D., and Wooldridrge, M. (2004). The dmars architecture: A specification of the distributed multi-agent reasoning system. *Autonomous Agents and Multi-Agent Systems*, 9(1-2):5–53.

[Falcone and Castelfranchi, 2000] Falcone, R. and Castelfranchi, C. (2000). Grounding autonomy adjustment on delegation and trust theory. *Journal of Experimental and Theoretical Artificial Intelligence*, 12(2):149–151.

[Ferber, 1999] Ferber, J. (1999). *Multi-Agent Systems - An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, Harlow, UK.

[Findler, 1991] Findler, N. V. (1991). Uttam sengupta: An overview of some recent and current research in the ai lab at arizona state university. *AI Magazine*, 12(3):22–29.

[FIPA, 1996] FIPA (1996). Foundation for intelligent physical agents. `http://www.fipa.org`. Software Standards for Heterogeneous and Interacting Agents and Agent-based Systems.

[FIPA, 2002] FIPA (2002). Fipa agent management specification. `http://www.fipa.org/specs/fipa00023/`. Document Nr.: SC00023J.

[FIPA CAL, 2001] FIPA CAL (2001). Fipa communicative act library specification. `http://www.fipa.org/specs/fipa00037/`. Document Nr.: XC00037H.

[FIPA CCL, 2001] FIPA CCL (2001). Fipa ccl content languages specification. `http://www.fipa.org/specs/fipa00009/`. Document Nr.: XC00009B.

[FIPA KIF, 2003] FIPA KIF (2003). Fipa kif content languages specification. `http://www.fipa.org/specs/fipa00010/`. Document Nr.: XC00010C.

[FIPA RDF, 2001] FIPA RDF (2001). Fipa rdf content languages specification. `http://www.fipa.org/specs/fipa00011/`. Document Nr.: XC00011B.

[FIPA SL, 2002] FIPA SL (2002). Fipa sl content languages specification. `http://www.fipa.org/specs/fipa00008/`. Document Nr.: SC00008I.

[Gomez-Sanz et al., 2004] Gomez-Sanz, J. J., Geravis, M.-P., and Weiss, G. (2004). A survey on agent-oriented software engineering research. In Bergenti, F., Gleizes, M.-P., and Zambonelli, F., editors, *Methodologies and Software Engineering for Agent Systems*, pages 33–64, Boston. Kluwer Academic Publishers.

[Hammer et al., 2005] Hammer, J., Langer, H., and Timm, I. J. (2005). Distributed knowledge management in the transportation domain. In Palwar, K. S., editor, *Innovations in Global Supply Chain Networks. Proceedings of the 10th International Symposium on Logistics (ISL 2005)*, pages 486–491, Lissabon.

[Hentze et al., 1993] Hentze, J., Brose, P., and Kammel, A. (1993). *Unternehmensplanung - Eine Einführung*. Bern, 2. edition.

[Herzog et al., 2001] Herzog, O., Tönshoff, H. K., Timm, I. J., and Woelk, P.-O. (2001). Adaptives und kooperatives Verhalten in Agentensystemen. In *Proceedings des 3. Kolloquiums des Schwerpunktprogramms "Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien". 10.-12. März in Hameln*, number 23/01, Bremen.

[Hill et al., 1998] Hill, W., Fehlbaum, R., and Ulrich, P. (1998). *Organisationslehre 1*. Verlag Paul Haupt, Bern, 5. auflage edition.

[Hülsmann and Berry, 2004] Hülsmann, M. and Berry, A. (2004). Strategic management dilemma: Ist nevessity in a world of diversity and change. In Lundin, R., editor, *roceedings of the SAM/IFSAM VIIth World Congress on Management in a World of Diversity and Change*, Göteborg, Sweden.

[Holler and Illing, 1990] Holler, M. J. and Illing, G. (1990). *Einführung in die Spieltheorie*. Springer, Berlin.

[Hoppe and Mesegeur, 1993] Hoppe, T. and Mesegeur, P. (1993). Vvt terminology: A proposal. *IEEE Expert*, 8(3):48–55.

[Kalenka and Jennings, 1997] Kalenka, S. and Jennings, N. R. (1997). Socially responsible decision making by autonomous agents. In Korta, K., Sosa, E., and Arrazola, X., editors, *Cognition, Agency and Rationality*, pages 135–149. Kluwer Academic Publishers.

[Kieser and Kubicek, 1992] Kieser, A. and Kubicek, H. (1992). *Organisation*. Walter de Gruyter, Berlin, 3. edition.

[Kirn, 1996] Kirn, S. (1996). *Foundations of Distributed Artificial Intelligence*, chapter Organizational Intelligence and Distributed Artificial Intelligence, pages 211–230. John Wiley & Sons, New York.

[Kirn, 2002] Kirn, S. (2002). Kooperierende intelligente softwareagenten. *Wirtschaftsinformatik*, 44(1):53–63.

[Kirn et al., 2006] Kirn, S., Herzog, O., Lockemann, P., and Spaniol, O., editors (2006). *Multiagent Engineering - Theory and Application in Enterprises*. Springer, Berlin.

[Knirsch and Timm, 1999] Knirsch, P. and Timm, I. J. (1999). Adaptive multiagent systems applied on temporal logsitics networks. In Muffatto, M. and Pawar, K. S., editors, *Logistics in the Information Age*, pages 213–218. SGE Ditoriali, Padova Italy.

[Küpper, 1997] Küpper, H.-U. (1997). *Controlling - Konzepte, Aufgaben und Instrumente*. 2. edition.

[Krempels et al., 2006] Krempels, K.-H., Scholz, T., Timm, I. J., Spaniol, O., and Herzog, O. (2006). *Multiagent Engineering - Theory and Application in Enterprises*, chapter Interaction Design, page zur Veröffentlichung angenommen. In [Kirn et al., 2006].

[Kreps, 1994] Kreps, D. M. (1994). *Mirkoökonomische Theorie*. Verlag Moderne Industrie, Landsberg/Lech, 1. edition.

[Langer et al., 2005] Langer, H., Gehrke, J. D., Hammer, J., Lorenz, M., Timm, I. J., and Herzog, O. (2005). Emerging knowledge management in distributed environments. In *Proceedings of the Workshop on Agent-Mediated Knowledge Management*, Workshop at the Fourth International Conference on Autonomous Agents and Multiagent Systems, 25-29 July, pages 14–26, Utrecht.

[Langer and Timm, 2004] Langer, H. and Timm, I. J. (2004). Distributed knowledge management in the transportation domain. In [Timm et al., 2004], pages 13–16.

[Liang and Zhu, 2005] Liang, H.-H. and Zhu, M.-L. (2005). Developing self-organized architecture solution according to model driven generative domain engineering. In Czap, H., Unland, R., Branki, C., and Tianfield, H., editors, *Self-Organization and Autonomic Informatics (I)*, volume 135 of *Frontiers in Artificial Intelligence and Application*, pages 97–104, Amsterdam. IOS Press.

[Liu, 2001] Liu, J. (2001). *Autonomous Agents and Multiagent Systems - Explorations in Learning, Self-Organization and Adaptive Computing*, chapter Self-Organized Autonomy in Multi-Agent Systems, pages 141–180. World Scientific, Singapore.

[Lorenz et al., 2005] Lorenz, M., Gehrke, J. D., Hammer, J., and Timm, I. J. (2005). Knowledge management to support situation-aware risk management in autonomous, self-managing agents. In Czap, H., Unland, R., Branki, C., and Tianfield, H., editors, *Self-Organization and Autonomic Informatics (I)*, volume 135 of *Frontiers in Artificial Intelligence and Application*, pages 114–128, Amsterdam. IOS Press.

[Lorenzen et al., 2006] Lorenzen, L., Scholz, T., Timm, I. J., Rudzio, H., Woelk, P.-O., Denkena, B., and Herzog, O. (2006). *Multiagent Engineering - Theory and Application in Enterprises*, chapter Integrated Process Planning and Production Control, page zur Veröffentlichung angenommen. In [Kirn et al., 2006].

[Luhmann, 1984] Luhmann, N. (1984). *Soziale Systeme, Grundrisse einer allgemeinen Theorie*. Suhrkamp, Frankfurt.

[Lux, 1995] Lux, A. (1995). Kooperationen mensch-maschine arbeit - ein modellierungsansatz und dessen umsetzung im rahmen des systems mekka. Dissertation, Technische Fakultät der Universität des Saarlandes, Saarbrücken.

[Menzies and Pecheur, 2004] Menzies, T. and Pecheur, C. (2004). Verification and validation and artificial intelligence. In *Advances of Computers 65*. Academic Press.

[Müller, 1997] Müller, H.-J. (1997). Towards agent systems engineering. *Data & Knowledge Engineering*, 23(3):217–245.

[Müller, 1993] Müller, J. (1993). *Verteilte Künstliche Intelligenz - Methoden und Anwendungen*. BI Wissenschaftsverlag, Mannheim.

[Moulin and Chaib-Draa, 1996] Moulin, B. and Chaib-Draa, B. (1996). An overview of distributed artificial intelligence. In O'Hare, G. M. P. and Jennings, N. R., editors, *Foundations of Distributed Artificial Intelligence*, pages 3–56, New York. John Wiley & Sons, Inc.

[Mylopoulos et al., 2001] Mylopoulos, J., Kolp, M., and Castro, J. (2001). Uml for agent-oriented software development: The tropos proposal. In Gogolla, M. and Kobryn, C., editors, *Proceedings of the 4th International Conference on the Unified Modeling Language, Modeling Languages, Concepts, and Tools*, number 2185 in LNCS, pages 422–441, Berlin.

[Negnevitsky, 2002] Negnevitsky, M. (2002). *Artificial Intelligence - A Guide to Intelligent Systems*. Addison-Wesley, Harlow, UK, 2 edition.

[Newell, 1962] Newell, A. (1962). Some problems of the basic organization in problem-solving programs. In *Proceedings of the second conference on Self Organizing systems*. Spartan Books.

[Nickles et al., 2002] Nickles, M., Rovatsos, M., and Weiss, G. (2002). A schema for specifying computational autonomy. In *Proceedings of the Third International Workshop "Engineering Societies in the Agents World" (ESAW'2002)*, Madrid, Spain.

[Padgham and Winikoff, 2004] Padgham, L. and Winikoff, M. (2004). *Developing intelligent agent systems: A practical guide*. Wiley.

[Pawlaszczyk et al., 2004] Pawlaszczyk, D., Dietrich, A. J., Timm, I. J., Otto, S., and Kirn, S. (2004). Ontolgies supporting cooperations in mass customization - a pragmatic approach. In Piotrowski, M., editor, *Proceedings of the International Conference on Mass Customization and Personalization - Theory and Practice in Central*

Europe, Rzeszow, Poland. University of Information Technology and Management (UITM).

[Pawlaszczyk et al., 2003] Pawlaszczyk, D., Timm, I., and Heine, C. (2003). Agentengestützte simulation und steuerung von logistikprozessen - der daisiy-ansatz. Arbeitsbericht 20, Technische Universität Ilmenau, Wirtschaftsinformatik 2, Ilemanu.

[Petsch, 2002] Petsch, M. (2002). Openness and security in the fipa standard. In Timm, I. J., Berger, M., Poslad, S., and Kirn, S., editors, *Proceedings of the International Workshop on Multiagnet Interoperability (MAI'02), 25TH German Conference on Artificial Intelligence (KI2002)*, September.

[Piller, 2003] Piller, F. T. (2003). *Mass Customization, Ein wettbewerbsstrategisches Konzept im Informationszeitalter*. Gabler, Wiesbaden.

[Pitt and Mamdani, 1999] Pitt, J. and Mamdani, A. (1999). Designing agent communication languages for multi-agent systems. In *Multi-Agent System Engineering - 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent Worlds, MAAMAW'99 in Valencia, Spain*, pages 102–114. Springer.

[Poslad and Charlton, 2001] Poslad, S. and Charlton, P. (2001). Standardizing agent interoperability: The FIPA approach. In Luck, M., editor, *Multi-Agent Systems and Applications: 9th ECCAI Advanced Course, ACAI 2001 and Agent Link's 3rd European Agent Systems Summer School, EASSS 2001, Prague, Czech Republic, July 2–13, 2001: Selected Tutorial Papers*, volume 2086 of *Lecture Notes in Computer Science and Lecture Notes in Artificial Intelligence*, pages 98–117, New York, NY, USA. Springer-Verlag Inc.

[Rao and Georgeff, 1991] Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents in a bdi-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR & R-91)*, pages 473–484. Morgan Kaufmann Publishers, San Mateo, CA.

[Rao and Georgeff, 1995] Rao, A. S. and Georgeff, M. P. (1995). BDI-agents: From theory to practice. In *Proceedings of the First International Conference on Multiagent Systems*, pages 312–319, San Francisco, California. AAAI-Press.

[Riedemann, 1997] Riedemann, E. H. (1997). *Testmethoden für sequentielle und nebenläufige Software-Systeme*. B.G. Teubner, Stuttgart.

[Rosenschein and Zlotkin, 1994] Rosenschein, J. S. and Zlotkin, G. (1994). *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. The MIT Press Series in Artificial Intelligence. The MIT Press.

[Rovatsos and Weiss, 2005] Rovatsos, M. and Weiss, G. (2005). Autonomous software. In Chang, S. K., editor, *Handbook of Software Engineering and Knowledge Engineering*, volume 3: Recent Advances, River Edge, New Jersey. World Scientific Publishing.

[Russell and Norvig, 1994] Russell, S. J. and Norvig, P. (1994). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

[Russell and Norvig, 2003] Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition edition.

[Sandholm, 1999] Sandholm, T. W. (1999). Distributed rational decision making. In Weiss, G., editor, *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*, pages 201–258. The MIT Press, Cambridge, Massachusetts.

[Scheer, 1995] Scheer, A.-W. (1995). *Wirtschaftsinformatik, Referenzmodelle für industrielle Geschäftsprozesse*. Springer, Berlin.

[Schegloff, 1987] Schegloff, E. A. (1987). Between macro and micro: Contexts and other connections. In Alexander, J., Giesen, B., and Münch, R., editors, *The Micro-Macro Link*, pages 207–237. University of California Press, Berkeley.

[Scholz et al., 2006] Scholz, T., Timm, I. J., Görz, G., Schiemann, B., and Herzog, O. (2006). *Multiagent Engineering - Theory and Application in Enterprises*, chapter Semantics for Agents, page zur Veröffentlichung angenommen. In [Kirn et al., 2006].

[Scholz et al., 2005a] Scholz, T., Timm, I. J., and Rudzio, H. (2005a). Versatile manufacturing infrastructures enabling adaptive supply chains. In Palwar, K. S., editor, *Innovations in Global Supply Chain Networks. Proceedings of the 10th International Symposium on Logistics (ISL 2005)*, pages 96–101, Lissabon.

[Scholz et al., 2005b] Scholz, T., Timm, I. J., and Spittel, R. (2005b). An agent architecture for ensuring quality of service by dynamic capability certification. In *Proceedings of the Third European Workshop on Multi-Agent Systems, EUMAS 2005*.

[Scholz et al., 2005c] Scholz, T., Timm, I. J., and Spittel, R. (2005c). An agent architecture for ensuring quality of service by dynamic capability certification. In *Proceedings der Third German Conference on Multiagent Systems Technologies (MATES 2005)*, volume 3550 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 130–140, Berlin. Springer.

[Scholz et al., 2004a] Scholz, T., Timm, I. J., and Woelk, P.-O. (2004a). Emerging capabilities in intelligent agents for flexible production control. In Ueda, K., Monostori, L., and Markus, A., editors, *Proceedings of the 5th International Workshop on Emergent Synthesis (IWES 2004)*, pages 99–105, Budapest.

[Scholz et al., 2004b] Scholz, T., Timm, I. J., and Woelk, P.-O. (2004b). Emerging capabilities in intelligent agents for flexible production control. In *Proceedings of the Second European Workshop on Multi-Agent Systems, EUMAS 2004*, Barcelona.

[Scholz-Reiter et al., 2004] Scholz-Reiter, B., Windt, K., and Freitag, M. (2004). Autonomous logistic processes: New demands and first approaches. In *Proceedings of the 37th CIRP International Seminar on Manufacturing Systems*, pages 357–362, Budapest, Hungaria.

[Schreiber and Wielinga, 1996] Schreiber, G. and Wielinga, B. J. (1996). Swi: Making knowledge technology work. *IEEE Expert*, 11(2).

[Singh et al., 1999] Singh, M. P., rao, A. S., and Georgeff, M. P. (1999). Formal methods in dai: Logic-based representation and reasoning. In [Weiss, 1999], pages 331–376.

[Smith, 1988] Smith, R. G. (1988). The contract net protocol: High-level communication and control in a distributed problem solver. In Bond, A. H. and Gasser, L., editors, *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, San Mateo.

[Smith and Davis, 1980] Smith, R. G. and Davis, R. (1980). Frameworks or cooperation in distributed problem solving. *IEEE Transactions on Systems, man and Cybernetics*, 11(1).

[Stuckenschmidt and Timm, 2002a] Stuckenschmidt, H. and Timm, I. J. (2002a). Adapting communication vocabularies using shared ontologies. In Cranefield, S., editor, *Proceedings of the Second International Workshop on Ontologies in Agent Systems, Workshop at 1st International Conference on Autonomous Agents and Multi-Agent Systems*, pages 6–12, Bologna, Italy.

[Stuckenschmidt and Timm, 2002b] Stuckenschmidt, H. and Timm, I. J. (2002b). Information integration on the semantic web. In Stuckenschmidt, H., Visser, U., and Wache, H., editors, *Ontology-Based Information Integration - Tutorial at the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW), Sigüenza, Spain,01.-04.10.2002*, volume Bericht, pages 9–13, Bremen. Mathematik und Informatik, Universität Bremen.

[Tecuci, 1998] Tecuci, G. (1998). *Building Intelligent Agents - An Apprenticeship Multistrategy Learning Thory, Methodology, Tool and Case Studies*. Academic Press, San Diego, California, USA.

[Timm, 2001a] Timm, I. J. (2001a). Dynamisches konfliktmanagement zur verhaltenssteuerung kooperativer agenten. In Jablonski, S., Kirn, S., Plaha, M., Sinz, E. J., vom Ende, A. U., and Weiß, G., editors, *Verteilte Informationssysteme auf der Grundlage von Objekten, Komponenten und Agenten: Proceedings der Verbundtagung VertIS 2001*, Rundbrief der GI-Fachgruppe 2.5.2 EMISA, Heft 3 und Rundbrief der GI-Fachgruppe 5.10 MobIS, 8. Jahrgang, Heft 2, pages 141–157, Bamberg. Universität Bamberg.

[Timm, 2001b] Timm, I. J. (2001b). Enterprise agents solving problems: The cobac-approach. In Bauknecht, K., Brauer, W., and Mueck, T., editors, *"Informatik 2001" - Tagungsband der GI/OCG Jahrestagung, 25.-28. September 2001*, pages 952–958, Universität Wien.

[Timm, 2003] Timm, I. J. (2003). *Dynamisches Konfliktmanagement als Verhaltenssteuerung Intelligenter Agenten*. Dissertation, Fachbereich Mathematik und Informatik, Universität Bremen, Bremen.

[Timm, 2004a] Timm, I. J. (2004a). *Adaption und Lernen von und in Organisationen*, chapter Selbstlernprozesse in der Agentenkommunikation, pages 103–127. VS Verlag für Sozialwissenschaften, Wiesbaden.

[Timm, 2004b] Timm, I. J. (2004b). *Dynamisches Konfliktmanagement als Verhaltenssteuerung Intelligenter Agenten*, volume 283 of *Dissertationen in der Künstlichen Intelligenz (DISKI)*. infix - AKA Verlagsgruppe, Köln.

[Timm and Dembski, 2005] Timm, I. J. and Dembski, N. (eingereicht im Oktober 2005). Implikationen selbststeuernder logistikprozesse - herausforderungen an die gestaltung und das management in der produktionslogistik. *Logistikmanagement*.

[Timm et al., 2004] Timm, I. J., Hellingrath, B., Kindsmüller, M. C., and Wache, H., editors (2004). *Proceedings of the First International Workshop on Applied Artificial Intelligence and Logistics - Special Focus on Mobile Solutions*, 27th German Conference on Artificial Intelligence (KI-2004) Workshop, Ulm. Universität Ulm.

[Timm et al., 2001a] Timm, I. J., Herzog, O., Tönshoff, H. K., and Woelk, P.-O. (2001a). Akzeptanz von agententechnologie in der industriellen anwendung - fortschritt durch transparenz und standardisierung? *Industrie-Management*, 17(6):13–16.

[Timm and Hillebrandt, 2006] Timm, I. J. and Hillebrandt, F. (2006). *Reflexive soziale Mechanismen*, chapter Autonomie und Institutionalisierung von Reflexion - Ein sozialer Mechanismus zum strategischen Management (zur Steuerung) von autonomen Softwaresystemen, page zur Veröffentlichung angenommen. VS Verlag für Sozialwissenschaften, Wiesbaden.

[Timm et al., 2001b] Timm, I. J., Knirsch, P., Herzog, O., Tönshoff, H. K., and Woelk, P.-O. (2001b). Mass customization als chance für kmu: Kooperative agenten für die informationslogistik. In Sebastian, H.-J. and Grünert, T., editors, *Logistik Management - Supply Chain Management und e-Business*, pages 401–409. Teubner, Stuttgart.

[Timm and Pawlaszczyk, 2002] Timm, I. J. and Pawlaszczyk, D. (2002). Pflichtenheft für einen prototypen zur entwicklung eines mass customization simulationssystems. Technical report, Technische Universität Ilmenau, Wirtschaftsinformatik 2, Ilmenau.

[Timm and Pawlaszczyk, 2005] Timm, I. J. and Pawlaszczyk, D. (2005). Large scale multiagent simulation on the grid. In Veit, D., editor, *Proceedings of the Workshop on Agent-based Grid Economics (AGE 2005) at the IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, Cardiff, UK.

[Timm et al., 2002] Timm, I. J., Schlieder, C., and Hermes, T. (2002). Autonomous behavior in multiagent systems. In *Interdisciplinary College 2002 - Focus Theme: Autonomy and Emotion*, pages 392–411, Günne, Möhnesee. GI.

[Timm et al., 2006a] Timm, I. J., Scholz, T., and Fürstenau, H. (2006a). *Multiagent Engineering - Theory and Application in Enterprises*, chapter From Testing to Theorem Proving, page zur Veröffentlichung angenommen. In [Kirn et al., 2006].

[Timm et al., 2006b] Timm, I. J., Scholz, T., and Herzog, O. (2006b). Capability-based emerging organization of autonomous agents for flexible production control. *Advanced Engineering Informatics Journal*, 1(Special Issue on Emergent Synthesis):Zur Veröffentlichung angenommen.

[Timm et al., 2006c] Timm, I. J., Scholz, T., and Knublauch, H. (2006c). *Multiagent Engineering - Theory and Application in Enterprises*, chapter The Engineering Process, page zur Veröffentlichung angenommen. In [Kirn et al., 2006].

[Timm et al., 2006d] Timm, I. J., Scholz, T., Krempels, K.-H., Herzog, O., and Spaniol, O. (2006d). *Multiagent Engineering - Theory and Application in Enterprises*, chapter From Agents To Multiagent Systems, page zur Veröffentlichung angenommen. In [Kirn et al., 2006].

[Timm et al., 2001c] Timm, I. J., Tönshoff, H. K., Herzog, O., and Woelk, P.-O. (2001c). Adaptive production control emerging from the cooperation of intelligent agents. In Katalinic, B., editor, *Intelligent Manufacturing & Automation: Focus on Precision Engineering - Proceedings of the 12th DAAAM International Symposium*, pages 485–486. DAAAM Internation Vienna, Jena.

[Timm et al., 2001d] Timm, I. J., Tönshoff, H. K., Herzog, O., and Woelk, P.-O. (2001d). Synthesis and adaption of multiagent communication protocols in the production engineering domain. In Butala, P. and Ueda, K., editors, *Proceedings of the 3rd International Workshop on Emergent Synthesis (IWES '01)*, pages 73–82, Bled, Slovenia. LAKOS, Fakulteta za strojnistvo.

[Timm and Woelk, 2003a] Timm, I. J. and Woelk, P.-O. (2003a). Ontology-based capability management for distributed problem solving in the manufacturing domain. In Schillo, M., editor, *Multiagent System Technologies - Proceedings of the First German Conference, MATES 2003, September, Erfurt*, Lecture Notes in Artificial Intelligence (LNAI), pages 168–179, Berlin.

[Timm and Woelk, 2003b] Timm, I. J. and Woelk, P.-O. (2003b). Ontology-based capability management for distributed problem solving in the manufacturing domain. In Schillo, M., Klusch, M., Müller, J., and Tianfield, H., editors, *Multiagent System Technologies - Proceedings of the First German Conference, MATES 2003, Erfurt, Germany*, volume 2831 of *Lecture Notes in Artificial Intelligence*, pages 168–179. Springer.

[Timm et al., 2001e] Timm, I. J., Woelk, P.-O., Knirsch, P., Tönshoff, J. K., and Herzog, O. (2001e). Flexible mass customisation: Managing its information logistics using adaptive co-operative multiagent systems. In Pawar, K. S. and Muffatto, M., editors, *Logistics and the Digital Economy. Proceedings of the 6th International Symposium on Logistics*, pages 227–232. Salzburg, Austria.

[Tönshoff et al., 2000a] Tönshoff, H. K., Herzog, O., Timm, I. J., and Woelk, P.-O. (2000a). Architektur eines agentenbasierten systems zur integrierten arbeitsplanung und fertigungssteuerung. *ZWF Zeitschrift für den wirtschaftlichen Fabrikbetrieb*, 95(12):601–604.

[Tönshoff et al., 2002a] Tönshoff, H. K., Herzog, O., Timm, I. J., and Woelk, P.-O. (2002a). Emerging virtual enterprises. In Ueda, K., editor, *Proceedings of the 4th International Workshop on Emergent Synthesis (IWES-02), May 9-10, Kobe University*, pages 217–224, Japan.

[Tönshoff et al., 2002b] Tönshoff, H. K., Herzog, O., Timm, I. J., and Woelk, P.-O. (2002b). Integrated process planning and production control based on the application of intelligent agents. In Teti, R., editor, *Proceedings of the 3rd CIRP International Seminar on Intelligent Computation in Manufacturing Engineering - ICME 2002, 3-5 July*, pages 135–140, Ischia,Italy.

[Tönshoff et al., 2000b] Tönshoff, H. K., Woelk, P.-O., Herzog, O., and Timm, I. J. (2000b). Modellierung intelligenter agenten in kooperativen multiagentensystemen - das wöchnerinnen szenario. In Kirn, S., editor, *Tagungsband "Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien", 24.-25. November 2000*, Ilmenau. Institut für Wirtschaftsinformatik, Technische Universität Ilmenau.

[Tönshoff et al., 2001a] Tönshoff, H. K., Woelk, P.-O., Herzog, O., and Timm, I. J. (2001a). Integrated process planning and production control - a flexible approach using co-operative agent systems. In Inasaki, I., editor, *Initiatives of Precision Engineering at the Beginning of a Milenium - Proceedings of 10th International Conference on Precision Engineering (ICPE), Yokohama, Japan, July 18th - 20th, 2001*, pages 857–861. Kluwer Academic Publishers, Boston.

[Tönshoff et al., 2002c] Tönshoff, H. K., Woelk, P.-O., Herzog, O., Timm, I. J., and Böß, V. (2002c). Agent-based in-house process planning and production control for enterprises in supply chains. In Sullivan, W. G., editor, *Proceedings of the 12th International Conference on Flexible Automation & Intelligent Manufacturing (FAIM), July 15th-17th*, pages 329–338, Dresden.

[Tönshoff et al., 2001b] Tönshoff, H. K., Woelk, P.-O., Timm, I. J., and Herzog, O. (2001b). Planning and production control using co-operative agent systems. In Dimitrov, D. and du Preez, N., editors, *Proceedings of the International Conference on Competitive Manufacturing (COMA '01)*, pages 442–449, Stellenbosch, South Africa.

[Toenshoff, 1999] Toenshoff, H. (1999). Dynamic modelling of process planning knowledge based on a shop-floor model. In *Proceedings of the 32nd CIRP International Seminar on Manufacturing Systems*, pages 497–504, Leuven, Belgium.

[van de Vliert, 1997] van de Vliert, E. (1997). *Complex Interpersonal Conflict Behaviour - Theoretical Frontiers*. Essays in Social Psychology. Psychology Press, East Sussex, UK.

[Visser et al., 2002] Visser, U., Stuckenschmidt, H., Schlieder, C., Wache, H., and Timm, I. (2002). Terminology integration for the management of distributed information resources. *Kuenstliche Intelligenz*, 16(1):31–34.

[Weiss, 1999] Weiss, G., editor (1999). *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge, Massachusetts.

[Weiss et al., 2005] Weiss, G., Fischer, F., Nickles, M., and Rovatsos, M. (2005). Operational modeling of agent autonomy: theoretical aspects and a formal language. In *Proceedings of the 6th International Workshop on Agent-Oriented Software Engineering (AOSE)*, Utrecht, The Netherlands.

[Weiss and Jakob, 2005] Weiss, G. and Jakob, R. (2005). *Agentenorientierte Softwareentwicklung*. Springer, Berlin.

[Wooldridge, 1999] Wooldridge, M. (1999). Intelligent agents. In [Weiss, 1999], pages 27–78.

[Wooldridge, 2002] Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. Wiley, West Sussex, UK.

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

[Wooldridge and Lomuscio, 2000] Wooldridge, M. and Lomuscio, A. (2000). Multiagent vsk logic. In *Proceedings of the Seventh European Workshop on Logics in Artificial Intelligence (JELIAI-2000)*. Springer-Verlag, Berlin.

[Wooldridge, 2000] Wooldridge, M. J. (2000). *Reasoning about Rational Agents*. The MIT Press, Cambridge, Massachusetts.