Available online at www.sciencedirect.com
ScienceDirect
Advanced Engineering Informatics 20 (2006) 247–259
ADVANCED ENGINEERING INFORMATICS
www.elsevier.com/locate/aei

Capability-based emerging organization of autonomous agents for flexible production control

Ingo J. Timm *, Thorsten Scholz, Otthein Herzog

University of Bremen, Center for Computing Technologies (TZI), Am Fallturm 1, D-28359 Bremen, Germany

Received 6 January 2006; accepted 19 January 2006

Abstract

Flexible structures in the manufacturing domain are of increasing concern. Nowadays, distributed systems for the coordination, management, and control of highly heterogeneous manufacturing structures are under research and development. However, their flexibility is restricted by two main aspects: On the one hand, in the design step of the system designers are implementing mostly static communication and reasoning skills for distributed software systems. On the other hand, knowledge on solutions provided by machine tools are implemented implicitly, i.e., the software system representing a machine tool can only reason on capabilities for a solution with respect to the capabilities implemented by the designer.

In this paper we will propose an innovative approach to dynamic capability management enabling software agents to reason on their capabilities and dynamically create capabilities as solutions.

Keywords: Emergent capabilities; Capability management; Multiagent systems; Flexible production control

1. Introduction

Flexible structures in the manufacturing domain are of increasing concern. Nowadays, distributed systems for the coordination, management, and control of highly heterogeneous manufacturing structures are under research and development. However, their flexibility is restricted by two main aspects: On the one hand, in the design step of the system designers are implementing mostly static communication and reasoning skills for distributed software systems. On the other hand, knowledge on solutions provided by machine tools are implemented implicitly, i.e., the software system representing a machine tool can only reason on capabilities for a solution with respect to the capabilities implemented by the designer. A promising approach to overcome these restrictions to the flexibility is the application of multiagent systems. Here, emergent behavior resulting from the coordination of autonomous problem solving units, i.e., agents, is used to reach more flexible behavior of the overall system. In multiagent architectures, emergent behavior became one of the most important criteria to ensure flexibility within multiagent systems [24]. The key elements of emergence in agent research can be found in the aspects: emergent properties, emergent organization, and transition from reactive to deliberative agents. The first aspect is focused on *emergent properties* as a large-scale effect of locally interacting agents: ''Emergent properties are often surprising because it can be hard to anticipate the full consequences of even simple forms of interaction'' [1]. Jacques Ferber's view is more centralized on *emergent organization*: ''Even societies considered as being complex such as colonies of bees or ants, should not necessarily be considered as individuals in their own right if we wish to understand their organization and the

* Corresponding author.
E-mail addresses: i.timm@tzi.de (I.J. Timm), scholz@tzi.uni-bremen.de (T. Scholz), herzog@tzi.uni-bremen.de (O. Herzog).
URL: http://www.tzi.de (I.J. Timm).

1474-0346/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.aei.2006.01.008

regulation and evolution phenomena prevailing there. In terms of multiagent systems, this means that an organization can emerge from the juxtaposition of individual actions, without its being necessary to define a specific objective (an element from the assembly O) which represent such an outcome'' [6]. 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*'' [34] arises within this theoretical basis.

A closer look on all three aspects supports the assumption that flexibility in interaction is the main reason for achieving global structures, e.g., dynamic organization, and intelligent behavior of a multiagent system as a whole. Following the discussion from Axelrod, emergence is associated with properties of a system, which become obvious during runtime of the system.

In the context of multiagent systems, emergent properties may result from the interaction between the agents. On the one hand the cooperation and coordination is following local optimization criteria (goals) and on the other hand it has to take a joint optimization criterion (goals of the multiagent system) into account. This collaboration should lead to a global optimization of the system (emergent effect). Thus, as communication is one of the main methodologies for achieving emergent behavior, adequate structure as well as contents are required. The main research in this context was focused on structural aspects, e.g., FIPA interaction protocol specification [7], or price-finding mechanisms, e.g., [24,17]. The second aspect of adequate communication allowing for emergent behavior of multiagent systems is addressed to the content resp. the object of negotiation. Here, the question is, how to implement cooperation between agents as flexible as possible. Cooperation is based on the mutual assumption that agents know if they can provide resp. request a service. If emergent organization is in question, the agents should be able to reason about their capabilities to identify tasks, in which solutions they can participate.

In the first section of this paper, we will discuss agents and their application in the manufacturing domain and identify the problems of static capability definitions. In the following section, we will define *capability management* and give its formal foundation based on the Discourse Agent architecture. Subsequently, we will introduce the ontology-based capability management algorithm *cobac** which is applied within a case study in the following section. In the final sections of this paper, we will discuss alternative methods for capability management and conclude with some summarizing remarks.

2. Rationale

Many enterprizes face radical changes within global production. Discovering new customers in new markets makes it necessary to offer highly customized and innovative products to meet customer's requirements. Within this context, companies have to solve the change towards flexible, demand driven production. New and more information has to be handled and a considerable speed-up of the development and manufacturing processes is needed. However, strong borderlines exist between process planning, production control and scheduling systems. The traditional approach of separating planning activities (e.g., process planning) from implementing activities (e.g., production control and scheduling) results in a gap between the involved systems. This gap results in loss of time and information. Thus, innovative concepts and methods for management and control of integrated information logistics, of production scheduling and of process planning are necessary. These concepts have to take several topics into consideration, which characterize the disadvantages of the current situation.

2.1. Autonomous agents

Since the early 90s, co-operative multiagent systems and intelligent agents are of increasing concern with respect to software engineering of large scale distributed systems [12]. However, MAS are not state-of-the-art for industrial applications. For an efficient and accepted application we should determine if MAS are adequate means for modeling organizations and temporary cooperation relations. Therefore it seems to be adequate to determine demands which must be met by the domain. In this context Müller proposes three requirements to be satisfied by the domain to ensure that a MAS can be applied fruitfully [15].

- *Natural distributivity.* When mapping a distributed domain to a model, it is essential to keep up the distributivity, or the distributivity lies within the task structure. On the other hand, the distributed structure should be an inherent property of the domain and not only of an artificial design.
- *Dynamic environment.* This demand does not only address changing data of the environment but also structural changes of the entire systems, e.g., machines which are added to or removed from the shop floor, partners who are giving up the cooperation or are joining it.
- *Flexible interaction.* The processes or objects which should be complemented with the help of a MAS are in need of complex interactions, e.g., they have to negotiate or exchange complex information. This usually requires the usage of well-defined semantics for the exchanged information items.

These requirements are corresponding to the criteria and demands described above. From this point of view it is adequate to use MAS for the process planning and production control domain. Especially the often world-wide distributed production, a constantly changing environment and conditions, as well as a high level of complexity of single

production processes and the information needed for the production are supporting this thesis.

Considering the application of agents, the question arises which kind of agents and MAS should be used. Regarding the different application areas of MAS it is obvious that finding an appropriate general definition of MAS covering all aspects is almost impossible [13]. MAS consist of distributed computational entities, the so-called agents. They are comparable to objects but are capable of sensing their environment and reacting according to the situation they perceive. Agents are goal-oriented, i.e., they receive tasks and pursue them subsequently. In contrast to objects, intelligent agents are rather autonomous in their behavior. Although they are pursuing goals, they are capable of choosing the concrete realization of the goal or of choosing the next goal to pursue. Thus, intelligent agents are implemented using mental states and explicit knowledge representation. Their autonomy is restricted to the scope granted by the instructing entity.

Agents which are situated in an environment should be able to co-operate for concurrent use of limited resources or for solving problems, e.g., scheduling, distributively. Therefore, the most important feature of a co-operative MAS is the social ability of each agent. The social behavior is implemented by the explicit definition of agent communication languages for co-ordination and co-operation. These languages and their protocols determine the expressive power and therefore the problem-solving abilities and the efficiency of the entire MAS. Furthermore, they must have knowledge of themselves as well as of the existence and competence of other agents. This enables them to act in open systems. In particular they need the capability to recognize agents entering or leaving the system. Internally, agents have their own views of their environment, and they need to adapt to and learn from changes that occur at runtime [32].

2.2. Agents in the manufacturing domain

Several applications of agent technology in manufacturing focus on digital marketplaces where intelligent agents act on behalf of enterprizes, customers or other organizations to achieve goals like to acquire a specific good for the smallest possible price. Since enterprizes began to consider several departments as individual profit centers, market-based coordination mechanisms became of increasing concern not only for inter-enterprizes relationship but also for internal processes of enterprizes like e.g., scheduling tasks at the shop floor. With respect to manufacturing of customized products in small lot sizes in batch job production, 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 is not a static one 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. Therefore, intelligent software agents turn out to be a suitable approach to develop dynamic, marketplace-based systems in the manufacturing domain.

Recent research projects deal with shop floor planning problems, e.g., by improving scheduling with respect to robustness and dynamic distributed environments. However, agent technology may be used to overcome existing traditional limitations in today's manufacturing systems, too [14]. For example, the *IntaPS* project implements a system of intelligent agents, which is capable of integrated process planning and production control to enhance the flexibility of planning at the shop floor. It is not bound to the restrictions of simple linear process plans any longer [28]. Since process plans are the result of agent communication, new alternative processing sequences can be found, e.g., in case of re-planning caused by unexpected machine breakdown. Order agents need knowledge about constraints related to the product's features and the necessary capabilities to manufacture these features. In addition, resource agents need knowledge about their capabilities. Thus, management of capabilities is important for negotiation of process plans.

3. Problem statement: capability management

Cooperative distributed problem solving (CDPS) is one of the main features of multiagent systems. Communication facilitates a very flexible way of organizing collaborative work among agents [2]. The coordination follows either the assumption of benevolence, i.e., agents are (implicitly) sharing common goals, or self-interest, i.e., agents pursue potentially conflicting local goals [5]. For multiagent systems in the manufacturing domain the assumption of benevolence is not suitable, esp. if not only processes within single enterprizes but also inter-enterprize cooperation are in question.

The process of CDPS can be divided into three stages [22]: (i) problem decomposition, (ii) sub-problem solution, and (iii) answer synthesis. In this paper, we are focusing on the first aspect: problem decomposition. An implicit assumption here is, that a society of agents is consisting of problem solving agents (PSA) and task agents (TA). Obviously, this is a very abstract view on the process of CDPS. In the manufacturing domain, we are dealing with a more complex situation: agents are representing manufacturing orders and machine tools – analogously, this can be extended to assembly or service processes. A simple approach for process planning would be linking manufacturing order

agents (MOA) to task agents and machine tool agents (MTA) to problem solving agents. For flexible process planning, a more sophisticated approach should enable machine tool agents to place suborders for load balancing purposes. Thus, a static link between PSA and MTA as well as between TA and MOA is insufficient and the introduction of roles is mandatory as it has been realized in the *IntaPS* project [28,4]. In this paper, we are using the terminology of PSA and TA to describe technical aspects; for comprehensive example purposes we are referring to MTA and MOA linked statically to PSA resp. TA.

Software systems managing information logistics within the manufacturing domain have to allocate resources to tasks. The level of autonomy within this process is eminent for flexible process planning and production control. In multiagent systems this allocation can be based on logical inference with the benefit of explicit semantics. As an example, in the *IntaPS* project, the task allocation is performed by knowledge-based classification using description logics. A common assumption here is, that the PSAs are able to decide whether or not they are capable of solving a requested task in respect to their capabilities. Implementations are using static mappings from capabilities to tasks resp. classes of tasks created at design time, mostly. Consequently, current approaches lack the flexibility to react on a changing environment where a different set of TAs provide new problems to solve for PSAs which require new capabilities.

Summarizing, capabilities are only under consideration during design time, i.e., agents are implemented for a set of problem-solving methods. Even in adaptive approaches to agent design and implementations the skills and capabilities of agents are addressed to in a static manner.

Requirements towards capabilities of agents are subject to change in dynamic environments, esp. if agents have the skill to form teams. In the case of dynamic team formation, the set of capabilities of a team can change significantly. Thus, we are stating, that explicit representation of as well as inference on capabilities and set of capabilities are in question to build flexible multiagent systems. In the following, we are addressing these problems with the term *capability management*.

In the following sections, we will introduce a formal foundation for capability management, the Discourse Agent Architecture, with the conflict based agent control (*cobac*) enabling agents to balance conflicts in goal selection. The architecture serves as the basis for the ontology based capability management based on the *cobac** algorithm proposed in the subsequent section.

4. Formal foundation: the Discourse Agent architecture

The Discourse Agent approach specifies an architecture for agent behavior, knowledge representation, and inferences for application in eBusiness, esp. in the manufacturing domain. This basic approach strictly separates internal and external aspects due to privacy and security issues. A three layer architecture is introduced 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., enterprize resource planning (ERP) systems and further information sources [2].

Nowadays, the communicator should be implemented with respect to standardization efforts, like FIPA [19]. In the case of our research project, the communicator is realized on top of a FIPA compliant agent toolkit (JADE, developed by CSELT S.p.A.[1]). The executor has to be implemented according to its directly connected resources, e.g., machine tools. While the communicator and executor layer is constructed in a straight forward manner, the design of the controller layer is more sophisticated implementing two innovative concepts: conflict-based agent control (*cobac*) and open, adaptive communication (*oac*).

The controller layer determines the behavior, strategy, and state of the agent. That means an agent behaves in the way the functions and procedures in the controller decide. It can only learn from experience acknowledged in the controller. The architecture presented here is based on the deliberative agent architecture BDI [21]. The formal foundation is introducing a new (multi-) modal logic, which integrates the formal approaches VSK-logic [35] for inter-agent behavior and the LORA-logic [33] for deliberative agent behavior. In the following, we are focusing on the internal behavior of an agent, where decisions on proposing for requested tasks are computed[2]. For further definitions, let Ω be the set of any well-formed formulas with respect to the grammatical definitions of this logic; $B* \subset \Omega$ is denoting the set of any possible belief. The main concepts needed for the definition of *Discourse Agents* are introduced in the following paragraphs:

Definition 1 (*DiscourseAgent*). A discourse agent is given by a 7-tuple:$Ag = \langle L, Act, see, reflect, decide, execute, l_0\rangle$, where $l_0 \in L*$ is denoting the initial state.

While L, Act, and *decide* are introduced in the next section, *see* (perception function, cf. [35]), *reflect* (knowledge revision function), and *execute* (action selection function) are not in the focus of this paper.

4.1. The local state

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.

Definition 2 (*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 $\mathrm{Act}^c = \{\alpha_0^c, \ldots, \alpha_m^c\}$ and $\mathrm{Act}^e = \{\alpha_0^e, \ldots, \alpha_n^e\}$ are denoting

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

[2] For a detailed introduction to the architecture, esp. the open, adaptive communication approach, refer to [26] or [27].

the communicative resp. executive actions of an agent. Together they are building the action potential $\mathrm{Act} = \mathrm{Act}^{e} \cup \mathrm{Act}^{c}$.

Following the definition of *Discourse Agent*, plans are tuples consisting of pre- and post-conditions, a set of available actions as well as mappings *status* and *select*.

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

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

The set Pln is denoting the set of all action plans of an agent.

A local state is defined using the mental categories *beliefs*, *desires*, and *intentions* as follows:

Definition 4 (*LocalState*). The local state is defined as 5-tuple: $L = \langle B, D, I, \mathrm{Plan}, \gamma \rangle$ if

- $B \subset B*$ is the set of current beliefs,
- $D \subset B*$ is the set of current desires,
- $I \subset D \times \mathrm{Pln}$ is the set of active intentions,
- $\mathrm{Plan} \subset \mathrm{Pln}$ is the set of available plans, and
- $\gamma : B* \times D \rightarrow \mathbb{R}$ is a mapping computing the relevance of a desire in the current situation.[3]

4.2. Conflict-based agent control

The main decision function within a Discourse Agent's controller 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 intention, which does not include intentions, which are not fitting the criteria of the pre-defined level of commitment (blind, single-minded or open-minded commitment [20]).

In the deliberation process an option is defined as a tuple of a desire and a plan (cf. intention, $o = \langle \mathrm{des}, \mathrm{plan} \rangle$, with $\mathrm{des} \in D$ and $\mathrm{plan} \in \mathrm{Pln}$). The option generation process is building 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 [33]; modifications on 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 is implementing the commitment to an intention and should ensure, that important and almost completed tasks will be finished first.

In the next step the options have to be filtered. The filtering is using 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 are receiving 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. Figs. 1 and 2 are showing the conceptual idea of synergy and conflict.

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 inter-personal conflict studies [29]. A conflict taxonomy is introduced in [26], where each pair of options is classified as a leaf in this taxonomy. 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 and two objectives pursuing conflicting post-conditions can be removed.

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

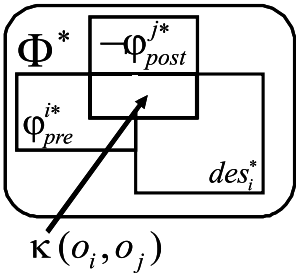

Fig. 1. Conflict value.

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

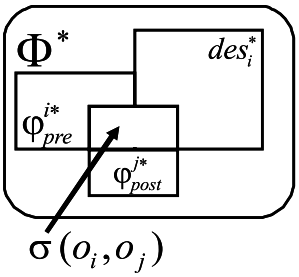

Fig. 2. Synergy value.

Definition 5 (*Decide*). The mapping: $\text{decide} : L* \to L*$ with $\text{decide}(\langle B, D, I, \text{Pln}, \gamma\rangle) \mapsto \langle B, D, I^*, \text{Pln}, \gamma^*\rangle$ is called conflict-based agent control if

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

5. Ontology-based capability management

On the basis of the previously introduced Discourse Agent architecture, we proposed a first step approach (*cobac*∗) to ontology-based capability management within PSAs in [25]. This approach is using taxonomic definitions of capabilities within an ontology and is realizing emerging capabilities.

5.1. Formal remarks on ontologies

In the definition of local states, we introduced desires as statements, which should become true in future states. Moving this approach to specification of problems, a problem is specified using statements, which describe pre- and post-conditions. We assume, that an explicit, formal ontology is provided for specification of the semantics of used symbols in these conditions. Ontologies can be formally specified as a 3-tuple, which consists at least of concepts, attributes, and relations.

Definition 6 (*Ontology*). Let C be a set of concepts, A a set of attributes, and R be a set of relations on these concepts, then a tuple $\text{Onto} = \langle C, A, R\rangle$ is called ontology.

In the following we are also assuming that an ontology Onto^t used for specification of problems contains at least one taxonomic relation without multiple inheritance ($r^t \in R$). As abbreviation we use length as a symbol for the amount of edges between two nodes, length(n) for the length from the root node to node n, and length(n, m) for the length from node n to node m. The mapping mscc is mapping a pair of concepts to the most specific common concept.

5.2. Capabilities

The capability management is using a problem description, an ontology, as well as an agent specified as a *Discourse Agent*. The agent receives a problem description and is performing the *cobac*∗ algorithm to decide how appropriate the agents capabilities are for the specified problem.

Following the definition of *Discourse Agent* the agent's capabilities are determined by its set of available plans. An action plan is describing a capability to transform a state which supports its pre-condition into a state supporting its post-condition. Taking ontological relations into account, an agent's capabilities could be extended, e.g., including super-concepts from taxonomic relations. The approach of capability management is not only extending capabilities but also problem descriptions. If a problem description is using a specific concept and the capability of an agent matches to the super-concept, the capability may be sufficient. The next paragraph is introducing a conflict-based approach to compute solutions as the above mentioned.

5.3. cobac*

The first step in CDPS for a problem solving agent is, to decide whether or not, it is capable of decomposing or solving the task. In this section we are proposing an algorithm to solve this decision using an ontology for the problem specification as well as an explicit specification of capabilities. This algorithm is using and extending the formal framework of *Discourse Agents*, presented in the last section.

As mentioned before, the Discourse Agent architecture is introducing the *cobac*-algorithm, which is a specialization of intention-selection within BDI agents. The main idea of *cobac* is to measure synergy and conflict within a pair of conflicting goals. Based on these measures, goals are combined, eliminated, or enhanced to minimize overall conflict and maximize synergy [26].

This algorithm is modified for capability management in cooperative distributed problem solving (*cobac**). The *cobac*-algorithm specified for the conflict-based agent control is using a specific conflict and synergy measure, which is based on partial correspondence and contrast of complex expressions. In opposite to this, the *cobac**-algorithm is introducing a conflict measure based on the evaluation of proximity and distance in an ontology, i.e., taxonomic relation. The goal of the algorithm is to find that capability, which is fitting best to the requested problem [25].

The *cobac* algorithm specified for the conflict-based agent control is using a specific conflict and synergy measure, which is based on partial correspondence and contrast of complex expressions. In opposite to this, the *cobac** algorithm is introducing a conflict measure based on the evaluation of proximity and distance in an ontology, i.e., taxonomic relation.

The goal of the algorithm is to find that capability resp. plan, which is fitting *best* to the requested problem. A requested problem is formalized as a tuple consisting of pre-conditions denoting the starting state and a post-condition denoting the goal state, such that problem $= \langle \varphi^p_{\text{pre}}, \varphi^p_{\text{post}} \rangle$. *cobac** is considering the mapping of problem to capability as a conflict and synergy assessment problem. The algorithm should return that capability, which has maximum synergy value and minimum conflict value. Therefore, the algorithm consists of the following three steps:

(1) Option generation;
(2) Conflict and Synergy assessment; and
(3) Option selection.

The computing of *cobac** is using a taxonomy r^t from the underlying ontology as reference relation. Starting the algorithm, a set of options is build, including all plans, which are available for the agent and which pre-conditions are supporting the pre-conditions of requested problem.

Definition 7 (*OptionGeneration*). A set of options Opt is constructed for a problem $\langle \varphi^p_{\text{pre}}, \varphi^p_{\text{post}} \rangle$ as specified by

$$\text{Opt} = \{\langle \varphi_{\text{pre}}, \varphi_{\text{post}}, A, \text{st}, \text{se} \rangle | \varphi_{\text{pre}} \rightarrow \varphi^p_{\text{pre}} \quad \text{and} \quad \langle \varphi_{\text{pre}}, \varphi_{\text{post}}, A, \text{st}, \text{se} \rangle \in \text{Plan}\}$$

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

Definition 8 (*Conflict and synergy*). For each option $o_i \in \text{Opt}$ and problem $p = \langle \varphi_{\text{pre}}, \varphi_{\text{post}} \rangle$ with a taxonomy $r^t \subset C^t \times C^t$, the conflict κ and synergy σ potential are calculated as follows:

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

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

The conflict and synergy values of a tuple of concepts c_i, c_j are determined by the distance in the taxonomic relation (cf. Section 3.1):

$$k(c_i, c_j) = \text{length}(c_i, c_j) \quad \text{and}$$
$$s(c_i, c_j) = \text{length}(\text{mscc}(c_i, c_j)).$$

The option selection is using these conflict and synergy potentials to select *most* appropriate option for the requested problem. This is determined by the quotient $\psi(p, o_i) = \frac{\kappa(p, o_i)}{\sigma(p, o_i)}$ of conflict and synergy potential, which is minimal if no conflict is found and maximal if no synergy is found. 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.

The algorithm presented here is using a taxonomic relation without multiple inheritance. This restriction can easily be overcome by introducing a slightly modified conflict and synergy assessment. Instead of using length of path – which assumes that there is exactly one path – longest path could be used for synergy assessment resp. shortest path for conflict assessment.

In the definition of Discourse Agents, we introduce desires as statements, which should become *true* in future states. Moving this approach to specification of problems, a problem is specified using statements, which describe pre- and post-conditions. We assume that an explicit, formal ontology is provided for specification of the semantics of used symbols in these conditions. Ontologies can be formally specified as a 3-tuple, which consists at least of concepts, attributes, and relations. In the following we are also assuming that an ontology used for specification of problems contains at least one taxonomic relation without multiple inheritance. As abbreviation we use *root* as the most abstract concept, length as a symbol for the amount of edges between two nodes, length(n) for the length from the root node to node n, and length(n, m) for the length from node n to node m. The mapping mscc(n, m) is mapping a pair of concepts to the most specific common concept.

The capability management is using a problem description, an ontology, as well as an agent specified as a Discourse Agent. The agent receives a problem description and is performing the *cobac** algorithm to decide how appropriate the agents capabilities are for the specified problem. Following the definition of Discourse Agent the agent's capabilities are determined by its set of available plans. An action plan is describing a capability to transform a state which supports its pre-condition into a state supporting its post-condition. Taking ontological relations into account, an agent's capabilities could be extended, e.g., including super-concepts from taxonomic relations. The approach of capability management is not only extending capabilities but also problem descriptions. If a problem description is using a specific concept and the capability of an agent matches to the superconcept, the capability may be sufficient. The next paragraph is introducing a conflict-based approach to compute solutions as the above mentioned.

The goal of the algorithm *cobac** is to find that capability resp. plan, which is fitting best to the requested problem. A requested problem is formalized as a tuple consisting of pre-conditions denoting the starting state and a post-condition denoting the goal state. *cobac** is considering the mapping of problem to capability as a conflict and synergy assessment problem. The algorithm should return that capability, which has maximum synergy value

and minimum conflict value. Therefore, the algorithm consists of the following three steps:

(1) Option generation;
(2) Conflict and synergy assessment; and
(3) Option selection.

The computing of *cobac** is using the taxonomy of the underlying ontology as reference relation. Starting the algorithm, a set of options is build, including all plans, which are available for the agent and which pre-conditions are supporting the pre-conditions of requested problem.

In the next step conflict and synergy between options and problem are assessed. Therefore, each option i is put into relation to the problem j, i.e., the conflict and synergy values are calculated for this pair of concepts. The conflict value is computed as distance from the most specific common concept to the concepts:

$$\text{conflict}(i,j) = \text{length}(\text{mscc}(i,j),j) + \text{length}(\text{mscc}(i,j),i) \quad (1)$$

Synergy is calculated analogously from the root concept to the most specific common subject:

$$\text{synergy}(i,j) = 2\text{length}(\text{root},\text{mscc}(i,j)) \quad (2)$$

The option selection is using these conflict and synergy potentials to select most appropriate option for the requested problem. This is determined by the quotient of $\frac{\text{conflict}(i,j)}{\text{synergy}(i,j)}$, which is minimal if no conflict is found and maximal if no synergy is found. The option with the lowest quotient 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. The algorithm presented here is using a taxonomic relation without multiple inheritances. This restriction can easily be overcome by introducing a slightly modified conflict and synergy assessment. Instead of using length of path – which assumes that there is exactly one path – longest path could be used for synergy assessment resp. shortest path for conflict assessment.

In this paper, we are focusing on the decision, if a problem-solving agent should propose for solving a requested problem or not. If an agent is proposing a solution to an approximate problem, the task agent has to decide, whether or not it is accepting this solution. We are proposing to use the conflict-based agent control *cobac* algorithm for identifying valid solutions with respect to its desire and the adaptive communication *oac* 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 is using the taxonomic relation and a maximum distance value. This value is determining if a solution provided with a specific capability is valid for the problem.

In this paper, we are focusing on the decision, if a problem-solving agent should propose for solving a requested problem or not. If an agent is proposing a solution to an approximate problem, the task agent has to decide, whether or not it is accepting this solution. We are proposing to use the conflict-based agent control *cobac* algorithm for identifying valid solutions with respect to its desire and the adaptive communication *oac* 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 is using the taxonomic relation and a maximum distance value mdv. This value is determining if a solution provided with the capability c_i is valid for the problem c_p: $(\text{length}(c_i, c_p) < \text{mdv})$.

6. Case study

Since the integration of process planning and scheduling is a very complex issue in the manufacturing domain, the basic concepts of an ontology-based capability management using problem solving agents are exemplified by a set of case studies based on the *IntaPS* approach. The *IntaPS* approach is jointly researched by the Center of Computing Technologies (TZI) and the Institute of Production Engineering and Machine Tools (IFW) and is a first approach towards agent-based integrating of process planning and production control (funded by the Deutsche Forschungsgemeinschaft, 2000–2006). In combination with TZI research in the collaborative research center on "Autonomous Cooperating Logistics Processes" (SFB 637) the Discourse Agent architecture for business applications has been developed and implemented. In the following, we will give a brief overview over the research project which serves as the basis for the case study for the application of capability management in this context.

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 Fig. 3.

The application of intelligent agents is based on two substantial components, which link together information systems of earlier stages of product development and the resources on the shop floor. This link is realized by decentralized planning on shop floor level and by rough level process planning. Decentralized planning units on shop-floor level are implemented by a multiagent system based on three different types of agents: resource agents, order agents, and service agents. Each relevant resource of the production system (machine tools, transportation devices, staff, virtual resources like information systems, etc.) and its real-world environment is represented by one resource agent within the virtual environment of the multiagent system platform. Resource agents provide local knowledge bases of the associated resources. Order agents are representing orders, which have to be manufactured. Due to its autonomy and pro-activity, an order agent is able to

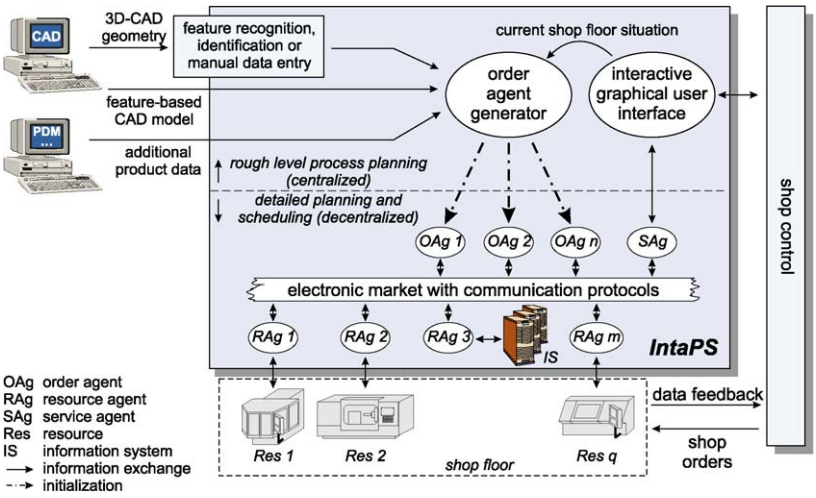

Fig. 3. The *IntaPS* approach.

recognize internal and external disturbances and to react appropriately. Service agents are used for human interaction, transparency and maintenance purposes.

The detailed process planning and scheduling takes place cooperatively within an electronic marketplace. Order agents and resource agents interact according to a ''three-phase- model''. Starting with a ''negotiation phase'' required manufacturing skills and due dates as well as capabilities and capacities are communicated.

Thus, order agents are able to identify appropriate resources which are capable to perform a specified 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. With respect to the description of the necessary manufacturing tasks, an approach based on technological constraints (e.g., described by manufacturing features according ''STEP-NC'' ISO 14649) is followed. Thereafter, suitable sequences of manufacturing operations are resulting 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 phase is called ''verification phase'' and ensures the feasibility of the detailed plan. The order agent examines continuously whether its detailed plan is executable under the current conditions. Changing situations (e.g., breakdown of a machine tool) cause order agents to analyze the consequences and to identify those parts of their detailed plans, which are affected. If necessary, the order agent enters a ''re-negotiation phase'' and tenders parts of the detailed plan for a new auction. The re-negotiation phase leads to an improved alternative detailed plan which substitutes the previous plan. Afterwards the verification phase is resumed and lasts until the order is finished. In addition to decentralized components of the multiagent system, centralized rough level planning pre-processes incoming data and generates a rough process plan. These data are geometrical or technological information about the product (e.g., from CAD systems), further organizational information related to products and orders (e.g., from PDM or ERP systems) as well as information concerning the current shop floor situation. Order agents are synthesized with respect to this information and initialized with a rough level process plan. This plan contains only information, which the agent is not able to recognize from the environment by itself (e.g., constraints between manufacturing operations). Thus, order agents obtain a maximum scope for the allocation of suitable resources and time slots for manufacturing. 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 distributed artificial intelligence and the use of emergent technologies for the integration of two units within an enterprize (process planning and production control), which were separated by a gap in information logistics in former times.

The case study for the application of capability management in the *IntaPS*-project deals with a simple manufacturing task of milling the cavity of an open step with curved surface shown in Fig. 4, which will be represented by a task agent (TA), respectively a manufacturing order agent (MOA).

Within traditional approaches, process planning of this manufacturing task will be carried out for example by a centralized process planning department. In addition, NC code generation may generate code assuming the same manufacturing strategy and similar process parameters for the whole curved surface (Fig. 4). However, real manufacturing systems consist of different resources with different capabilities. Thus, it is recommended to apply different planning strategies for different ''sub-tasks'' of the manufacturing task in an environment with distributed

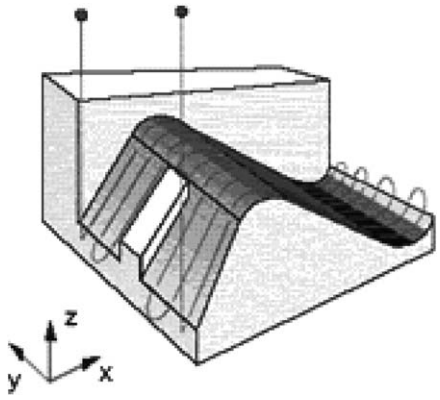

Fig. 4. Manufacturing task "milling of a part with curved surface" of the manufacturing order.

problem solving agents. In this case, decomposition of the whole manufacturing task into sub-tasks is a pre-requisite for further planning and requires a well-structured view on the manufacturing task. A STEP-NC-based representation of manufacturing objects is used for this purpose (Fig. 5).

STEP-NC is based on the well-established product data and representation standard STEP (ISO 10303), which evolved to one of the most important standards for system-independent product data exchange in engineering (e.g., automotive and aerospace industry). STEP-NC itself is a new standard for future, flexible and object-oriented NC code generation. It is currently available as a draft international standard (DIN ISO 14649) for milling, turning, and EDM. Since STEP and STEP-NC utilize very similar and sometimes the same methodologies (e.g., ISO 10303-11 EXPRESS modeling language for specification of the basic data model entities), these two standards are highly interoperable. Thus, it is possible to link geometrical entities of STEP-based CAD models from the product design domain with corresponding manufacturing objects represented by STEP-NC entities in an easy manner. Furthermore, these standards serve as a basis for ontology design in the *IntaPS* multiagent system. Thus, *IntaPS* agents are able to negotiate about STEP-represented planning problems.

With respect to the case study, a structured view on the manufacturing task is given by a STEP-NC-based set of manufacturing objects, e.g., a set of "toolpath" objects and their segments depicting the necessary movements of the milling tool in order to machine the surface. Since the machined surface is curved, each toolpath segment is characterized by different geometrical and cutting conditions. Thus, different strategies and machine parameters are necessary for optimal processing of the surface. Based on this classification, resource agents are able to calculate individual manufacturing parameters through which the quality of the generated schedule increases with respect to estimated manufacturing times, which will lead to a more robust and reliable production planning.

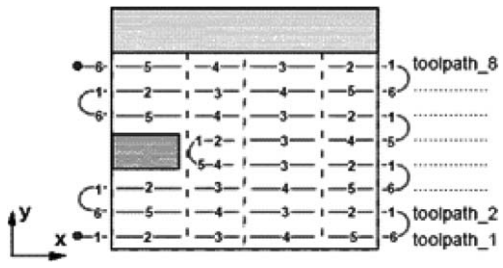

Fig. 5. Structured view on the manufacturing task using manufacturing objects based on STEP-NC.

This example emphasizes how to structure a manufacturing task. Since STEP-NC is able to represent not only some surfaces, but also whole work pieces as well as STEP offers a data model to represent complex assemblies and product structures, this approach is suitable to be applied to even more complex manufacturing tasks. Nevertheless, the method of decomposing the whole task remains the same. Thus, a manufacturing order agent is able to search for suitable problem solving agents (PSA), respectively resource agents, for different manufacturing object. Some of these manufacturing objects may be processed by a single resource with corresponding capabilities for each manufacturing object. Other manufacturing objects may require a combination of capabilities. Hence, ontology-based capability management will improve distributed planning activities as well as team formation processes (e.g., to form virtual manufacturing cells as teams of resources with combined capabilities and planning abilities) in *IntaPS* and similar multiagent-based applications.

7. Related work

There are different approaches to address to capability management from an AI perspective. We will discuss the most suitable techniques from knowledge representation, configuration, and planning.

In the field of knowledge representation and inference there is a broad variety of languages and inference algorithms available. For selection of a specific language, it is important to consider the requirements of the application: an autonomous agent should be able to reason about its capabilities for problem solving. Furthermore, in teams, agents have to communicate and negotiate to determine the team's capabilities. In consequence, representation languages are in question only, if inferences on an automated basis are available. In the context of the semantic web, ontologies have proven their use as an adequate means for formal representation and reasoning about knowledge. The main inferences within ontologies are based on classification for which efficient implementations exists (e.g., [10] or [11]). However, efficiency of concrete applications is depending on domain restrictions. Gruber defines an ontology as a explicit specification of a conceptualization [8]. E.g., an ontology covers all objects of interest, together

with definitions for the meaning of each of the terms in a given domain. Formal axioms are used to enforce constraints on the entities to describe their behavior. More precise the ontology represents the intended meaning of a formal vocabulary related to a particular conceptualization of the world [9].

Using ontologies as a basic technique for capability management, there are promising approaches in the field of semantic mediation [31] to support emergent capabilities, i.e., the creation of capabilities in teams. The key idea here is that generic capabilities are represented in a terminology. For each generic capability, at least one abstract partial order plan for solving the problem in question is associated. These can be composed to more complex capabilities within an agent. In teams of agents capabilities are created by negotiating about complex agent capabilities and combining them analogous to the reasoning process within an agent.

An extension of the approach outlined above can be found in the use of knowledge-based configuration. Configuration techniques use different inferences than ontologies. Instead of classification, most configuration approaches use constraint satisfaction to reason about knowledge. A combination of both (classification and constraint satisfaction) can be found in the heuristic structure-based configuration developed in the projects TEX-K and PROKON (e.g., [3]). The key requirement for the use of this approach is the component structure of the domain. Especially in the manufacturing domain the component structure is commonly found. This approach is based on three different knowledge representations:

(1) The knowledge about the objects of the domain is represented in an ontology according to the taxonomic and partonomic relations and the relevant attributes.
(2) Dependencies between objects and their relations and attributes are represented with constraints by a meta-constraint modeling language.
(3) Knowledge about the control process i.e., order of configuration steps, calculation methods and the priority among them as well as the use of conflict resolution methods is declaratively described by strategies.

Analogously to the prior approach, the ontology serves as the representation of taxonomic and partonomic relations between capabilities. Additionally, constraints are used to model explicit knowledge about the combination of capabilities. This enables the agent to reason about capabilities more efficiently as illegal combinations are forbidden through the constraint guards.

A different approach can be found in the usage of planning techniques. As introduced in the section on Discourse Agents, most agent approaches are using explicit or implicit representation of plans. If there is no explicit representation of agents' capabilities, the details on pre- and post-conditions of plans can be applied as an implicit capability representation.

Hierarchical task networks (HTN) are a state-of-the-art representative of partial order planning. In the context of capability management, HTN are plans which consist of subplans. Each of the subplans can be considered as a capability while actions would represent generic capabilities. If the capabilities defined directly are not sufficient to solve a requested task, plans have to be integrated. Thus, the task of combining capabilities can be solved through plan integration which is a major field in current planning research.

8. Conclusion and future work

The integration of process planning and production control in the *IntaPS* multiagent system is based on the allowing resource agents to infer about their own capabilities. Since this proved to be a successful step to overcome the gap between process planning and production control, our next steps were to generalize this approach for agent capabilities. We introduced the problem of capability management as restriction to the flexibility of multiagent systems in the manufacturing domain. On this basis, we discussed alternative approaches to enabling capability management within software agents from an AI perspective. We introduced the *cobac** approach for ontology-based capability management. *cobac** enables dynamic organization of agents by subsumption-based match-making within ontologies. The problem with using ontologies is that the representation of pre- and post-conditions of internally used plans has to be matched to problem descriptions. Most often, the heterogeneity of the ontologies of tasks and solutions leads to similar problems as known from the intelligent information integration community. However, integration of ontologies into internal planning processes of the agent is needed. By extending the representation of formal capabilities with constraint knowledge as in the structure based configuration, the integration of pre- and post-conditions appears to become tractable, yet a new set of problems arises. For the combination and recombination of capabilities and associated constraints, single constraints will have to be merged for the combination of simple capabilities and whole constraint nets will have to be integrated for more complex capabilities. Thus, a different approach should be used. With respect to the planning approaches, there are problems in combining capabilities and using combined – emerging – capabilities within planning processes. The approaches from planning are enabling a capability management bottom-up, i.e., from generic actions to high-level behavior. In opposite, an ontology approach is more a top-down approach starting from the problem description.

We have described an approach for capability management in cooperative distributed problem solving. The explicit integration of ontologies for deciding whether or

not an agent is capable of solving a problem enables an agent to propose a solution for a similar problem. The use of taxonomies as basic concepts is a first step, only. We are assuming that, there is a benefit of using a simple conflict measure in contrast to use no measure at all. The main benefit of the algorithm proposed here is, that a static link from a specific task to a specific capability is no longer needed. These enables multiagent systems to solve problems using alternatives in solving (sub-) tasks. In the manufacturing domain, the system is more flexible in planning and scheduling as more alternatives become available and the mapping from feature to machine tool can be solved dynamically. Furthermore, the algorithm is simple, such that an efficient implementation is possible.

The problem of capability management is still an open problem and will probably not be solved soon. One of the main restrictions of the approach proposed here lays in the assumption, that only unique and shared ontologies between agents exists. Using multiple ontologies may be necessary in real distributed scenarios, but is leading to the problem of equality or equivalence of concepts. Further shortcomings arise from using a simple length evaluation (amount of edges), such that taxonomic relations have to be designed carefully as its quality is corresponding to accuracy of this algorithm.

In the intelligent information integration as well as the semantic web community multiple approaches of estimating similarity of concepts in ontologies or semantic matchmaking are under research, e.g., [16,30,18]. We are going to integrate approaches of approximate reasoning, as applied to adaptation of communication vocabularies [23]. Additionally, we will focus on enhancing the capability management in our future research by integrating techniques from plan integration, ontologies, and structure-based configuration.

The use case shows high potential for application of ''weak'' mapping of capabilities to problems in the domain of distributed systems in cooperative manufacturing settings. Of course, more research and sophisticated algorithms for conflict assessment but also for building equivalence relations within the task agents are needed.

Acknowledgements

The authors would like to thank the reviewers for their valuable and helpful remarks. This research is partially funded by the German Research Foundation (DFG) as the Collaborative Research Centre 637 Autonomous Cooperating Logistic Processes: A Paradigm Shift and its Limitations (SFB 637). Additional Information can be found at http://www.sfb637.uni-bremen.de/. Parts of the presented work are funded by the Deutsche Forschungsgemeinschaft (DFG) within the projects He 989/5-2 and To 56/149-2 as part of the Priority Research Program 1083 ''Intelligent Agents and Realistic Commercial Application Scenarios''.

References

[1] R.M. Axelrood, The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration. Princeton Studies in Complexity, Princeton University Press, 1997.
[2] W. Conen, G. Neumann, Coordination Technology for Collaborative Applications. Number 1364 in Lecture Notes in Artificial Intelligence, Springer, Berlin, 1998.
[3] R. Cunis, A. Guenter, I. Syska, H. Peters, H. Bode, Plakon – an approach to domain-independent construction, in: IEA/AIE '89: Proceedings of the 2nd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, ACM Press, New York, NY, USA, 1989, pp. 866–874.
[4] B. Denkena, P.-O. Woelk, O. Herzog, T. Scholz, Integration of process planning and scheduling using intelligent software agents, in: Proceedings of the 36th CIRP International Seminar on Manufacturing Systems ''Progress in Virtual Manufacturing Systems'', Saarbruecken, 03–05.06.2003, Saarland University, 2003, pp. 537–544.
[5] E.H. Durfee, J.S. Rosenschein. Distributed problem solving and multi-agent systems: comparisons and examples, in: Proceedings of the 13th International Distributed Artificial Intelligence Workshop, 1994, pp. 94–104.
[6] J. Ferber, Multi-Agent Systems – An Introduction to Distributed Artificial Intelligence, Addison-Wesley, Harlow, UK, 1999.
[7] Fipa interaction protocol library specification. Available from: <http://www.fipa.org/specs/fipa00025/>, 2001, Document Nr.: XC00025E.
[8] T. Gruber, A translation approach to portable ontology specifications, Knowledge Acquisition 5 (2) (1993) 199–220.
[9] N. Guarino, Formal ontology and information systems, in: N. Guarino (Ed.), Formal Ontology in Information Systems, IOS Press, 1998.
[10] V. Haarslev, R. Moeller, Racer system description, in: Proceedings of the International Joint Conference on Automated Reasoning, IJCAR'2001, June 1823, Siena, Italy, Springer, 2001.
[11] I. Horrocks, U. Sattler, S. Tobies, Practical reasoning for expressive description logics, in: H. Ganzinger, D. McAllester, A. Voronkov (Eds.), Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99), Number 1705 in Lecture Notes in Artificial Intelligence, Springer, Berlin, 1999, pp. 161–180.
[12] N.R. Jennings, M.J. Wooldridge, Agent Technology. Foundations, Applications, and Markets, Springer, Berlin, 1998.
[13] P. Knirsch, I.J. Timm, Adaptive multiagent systems applied on temporal logistics networks, in: M. Muffatto, K.S. Pawar (Eds.), Logistics in the Information Age, SGE Ditoriali, Padova, Italy, 1999, pp. 213–218.
[14] V. Marek (Ed.), Knowledge and Technology Integration in Products and Services, Proceedings Balancing Knowledge and Technology in Product and Service Life Cycle (BASYS'02), Kluwer Academic Publishers, Cancun, Mexico, 2002.
[15] H.-J. Mueller, Towards agent systems engineering, Data and Knowledge Engineering 23 (3) (1997) 217–245.
[16] N.F. Noy, M.A. Musen, Evaluating ontology mapping tools: requirements and experience, in: Proceedings of the Workshop on Evaluation of Ontology Tools at EKAW'02 (EON2002), Siguenza, Spain, October 2002.
[17] D. Ouelhadj, S. Petrovic, P.I. Cowling, A. Meisels, Inter-agent cooperation and communication for agent-based robust dynamic scheduling in steel production, Advanced Engineering Informatics 18 (3) (2004) 161–172.
[18] M. Paolucci, T. Kawamura, T.R. Payne, K. Sycara, Semantic matching of web services capabilities, in: Proceedings of the 1st International Semantic Web Conference (ISWC2002), 2002.
[19] S. Poslad, P. Charlton, Standardizing agent interoperability: the FIPA approach, in: M. Luck (Ed.), 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, 2–13 July 2001: Selected Tutorial Papers, Springer-Verlag Inc, New York, NY, USA, 2000, pp. 98–117.
[20] A.S. Rao, M.P. Georgeff, in: Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR & R-91), Chapter Modeling Rational Agents in a BDI-architecture, Morgan Kaufmann Publishers, San Mateo, CA, 1991, pp. 473–484.
[21] A.S. Rao, M.P. Georgeff, BDI-agents: from theory to practice, in: Proceedings of the First International Conference on Multiagent Systems, San Francisco, 1995.
[22] R.G. Smith, R. Davis, Frameworks or cooperation in distributed problem solving, IEEE Transactions on Systems, Man and Cybernetics 11 (1) (1980).
[23] H. Stuckenschmidt, I.J. Timm, Adapting communication vocabularies using shared ontologies, in: S. Cranefield (Ed.), Proceedings of the Second International Workshop on Ontologies in Agent Systems, Workshop at 1st International Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy, 15–19 July 2002, pp. 6–12.
[24] I.J. Timm, H.K. Toenshoff, O. Herzog, P.-O. Woelk, Synthesis and adaptation of multiagent communication protocols in the production engineering domain, in: P. Butala, K. Ueda (Eds.), Proceedings of the 3rd International Workshop on Emergent Synthesis (IWES'01), Bled, Slovenia, 12–13 March 2001, LAKOS, Fakulteta za strojnistvo, pp. 73–82.
[25] I.J. Timm, P.-O. Woelk, Ontology-based capability management for distributed problem solving in the manufacturing domain, in: M. Schillo (Eds.), Multiagent System Technologies – Proceedings of the First German Conference, MATES 2003, September, Erfurt, Lecture Notes in Artificial Intelligence (LNAI), Berlin, 2003, pp. 168–179.
[26] I.J. Timm, Enterprise agents solving problems: the cobac-approach, in: K. Bauknecht, W. Brauer, Th. Mueck (Eds.), Informatik 2001 – Tagungsband der GI/OCG Jahrestagung, 25–28 September 2001, Universitaet Wien, 2001, pp. 952–958.
[27] I.J. Timm, Dynamisches Konfliktmanagement als Verhaltenssteuerung Intelligenter Agenten, Volume 283 of Dissertationen in der Kuenstlichen Intelligenz (DISKI). infix – AKA Verlagsgruppe, Koeln, 2004.
[28] H.K. Toenshoff, P.-O. Woelk, I.J. Timm, O. Herzog, Planning and production control using co-operative agent systems, in D. Dimitrov, N. du Preez (Eds.), Proceedings of the International Conference on Competitive Manufacturing (COMA'01), Stellenbosch, South Africa, 31st January–2nd February 2001, pp. 442–449.
[29] E. van de Vliert, Complex Interpersonal Conflict Behaviour – Theoretical FrontiersEssays in Social Psychology, Psychology Press, East Sussex, UK, 1997.
[30] U. Visser, H. Stuckenschmidt, C. Schlieder, H. Wache, I. Timm, Terminology integration for the management of distributed information resources, Kuenstliche Intelligenz 16 (1) (2002) 31–34.
[31] H. Wache, Semantische Interpretation fuer heterogene Informationsquellen, volume 261 of DISKI – Dissertationen zur Kuenstlichen Intelligenz. infix, Koeln, 2003.
[32] G. Weiss (Ed.), Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence, The MIT Press, Cambridge, MA, 1999.
[33] M.J. Wooldridge, Reasoning about Rational Agents, The MIT Press, Cambridge, MA, 2000.
[34] M. Wooldridge, in: Weiss [32], Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence, Chapter Intelligent Agents, 1999, pp. 27–78.
[35] M. Wooldridge, A. Lomuscio, in: Proceedings of the Seventh European Workshop on Logics in Artificial Intelligence (JELIAI-2000), Chapter Multi-Agent VSK Logic, Springer-Verlag, Berlin, 2000.