Ontology-Based Capability Management for Distributed Problem Solving in the Manufacturing Domain

Ingo J. Timm[1] and Peer-Oliver Woelk[2]

[1] Technische Universitaet Ilmenau, Institute of Information Systems,
P. O. Box 10 05 65, 98684 Ilmenau, Germany
itimm@acm.org
[2] University of Hannover, Institute of Production Engineering and Machine Tools,
Schlosswender Str. 5, 30159 Hannover, Germany
woelk@ifw.uni-hannover.de

Abstract. Providing services within multiagent systems, an agent has to register itself with a distinct description of its main capabilities in yellow page services. If another agent is requesting to solve a specific task, it has to be decided whether or not the requested agent is capable of performing the task successfully. We are assuming that task requirements as well as capabilities are specified using ontologies. Decision is easy if the concepts of requested task requirements are directly mapping to concepts of provided capabilities. However, concept inequality may occur. Especially in production engineering with its increasing concern of knowledge about sophisticated manufacturing processes, relying on simple concept equality is not suitable to fulfill demands of current industrial applications. Thus, enhanced methods like ontology-based capability management presented in this paper have to be established to address this problem. For the case of indifferent concepts we are introducing a conflict-based approach for capability negotiation as well as an application scenario for this approach in the manufacturing domain.

1 Introduction

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 [1]. 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 [2]. For multiagent systems in the manufacturing domain the assumption of benevolence is not suitable, esp. if not only processes within single enterprises but also inter-enterprise cooperation are in question.

The process of CDPS can be divided into three stages: (i) problem decomposition, (ii) sub-problem solution, and (iii) answer synthesis [10]. In this paper, we are focusing on the first aspect: problem decomposition. An implicit assumption

M. Schillo et al. (Eds.): MATES 2003, LNAI 2831, pp. 168–179, 2003.
© Springer-Verlag Berlin Heidelberg 2003

here is, that each of the problem solving agent has knowledge on its capabilities. A more sophisticated approach would rather ask the question: *How does a problem-solving agent knows whether or not it is capable of solving the requested problem?*

1.1 Distributed Problem Solving in the Manufacturing Domain

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 to 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.

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. Hence, this problem may be solved by methods of CDPS. 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 [3]. 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 [13]. 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.

1.2 Problem Decomposition

Problem decomposition is the first stage in CDPS. The coordination of agents in this approach is based on message passing, which fits well into the application domain under the assumptions made above. The contract net protocol is a common representative for task decomposition algorithm in a message passing setting [9]. The problem solving process may be described step-wise: in each step, the problem solving agent provides 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. Thus, an distributed task decomposition algorithm can be realized. Of course, a problem solving agent has to be able to decide whether or 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 whole in a product, but in some cases and with specific machine tools milling may produce an equivalent result. Nevertheless, both actions are not equal. In this context, the mapping of capability to task and vice versa, which is not based on simple concept or string equality is an open issue.

2 Discourse Agent

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. enterprise resource planning (ERP) systems and further information sources [12].

Nowadays, the communicator should be implemented with respect to standardization efforts, like FIPA [6]. 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 [8]. The formal foundation is introducing a new (multi-) modal logic, which integrates the formal approaches VSK-logic [16] for inter-agent behavior and the LORA-logic [17] for deliberative agent behavior. In the following, we are

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

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; $\mathcal{B} \subset \Omega$ is denoting the set of any possible belief. The main concepts needed for the definition of *Discourse Agents* are introduced in the following paragraphs:

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

While L, Act, and $decide$ are introduced in the next section, see (perception function, cf. [16]) , $reflect$ (knowledge revision function), and $execute$ (action selection function) are not in the focus of this paper.

2.1 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.

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* $Act^c = \{\alpha_0^c, \ldots, \alpha_m^c\}$ *and* $Act^e = \{\alpha_0^e, \ldots, \alpha_n^e\}$ *are denoting the communicitave resp. executive actions of an agent. Together they are building the action potential* $Act = Act^e \cup Act^c$.

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*.

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, if*

- $status : B \times \varphi_{pre} \times \varphi_{post} \mapsto x$, *with* $x \in \mathbb{R}_0$ *and* $B \in \mathcal{B}$, *is a mapping denoting the execution status of the plan and*
- $select : status \times \mathcal{B} \to A$ *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:

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

[2] For a detailed introduction to the architecture, esp. the open, adaptive communication approach, refer to [12].

- $B \subset \mathcal{B}$ *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*[3].

2.2 Conflict-Based Agent Control (*cobac*)

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) [7].

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 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 [17]; 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. Figures 1 and 2 are showing the conceptual idea of synergy and conflict.

[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.

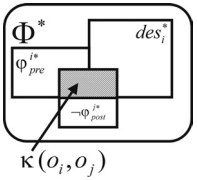


Fig. 1. Conflict Value

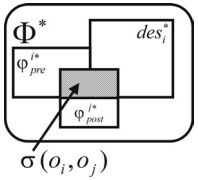


Fig. 2. Synergy Value

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 inter-personal conflict studies [14]. A conflict taxonomy is introduced in [12], 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:

5 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 if*

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

3 Task Decomposition Using Ontologies

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 capabilites. This algorithm is using and extending the formal framework of *Discourse Agents*, presented in the last section.

3.1 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.

6 Definition (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 $Onto = \langle C, A, R \rangle$ 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.

3.2 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.

3.3 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 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 $problem = \langle \varphi^p_{pre}, \varphi^p_{post} \rangle$. *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.

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 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 $\varphi^p_{post}, \varphi^i_{post}$. 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.

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

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

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

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):*

$k(c_i, c_j) = length(c_i, c_j)$ *and*
$s(c_i, c_j) = length(mscc(c_i, c_j))$.

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.

3.4 Accepting an Approximate Problem Solver

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: $(length(c_i, c_p) < mdv)$.

4 Scenario: The Gear Wheel

As described above, ontologies resp. a taxonomic relations are used as a basis for capability. In order to illustrate this approach, two quite simple ontologies will be used. The presented example ontologies reflect a small cut-out of the domain only and are tailored in order to represent the concepts needed for the following use cases. Figure 3 is showing an ontology for manufacturing features, the figure 4 focuses on manufacturing processes.

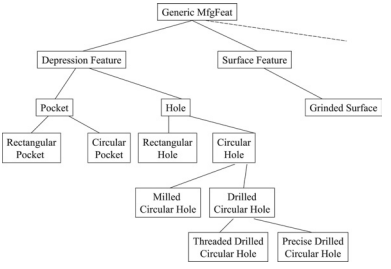


Fig. 3. Ontology of Manufacturing Features

4.1 Use Case A: An Order on Its Way to a Machine

The first use case illustrates how an order agent determines the sequence of necessary manufacturing operations based on capabilities offered by resource agents. It is assumed, that an order agent is requesting the manufacturing of a gear wheel by a cutting process out of forged raw material. In this context, an appropriate process for the tooth flanks is needed, but several cutting processes are available. However, conflicts may occur between different processes since the appropriate cutting process depends on the applied forging process.

Conventional forging processes lead to raw parts with rough surfaces and large allowance (amount of material to removed in final manufacturing steps). Since grinding processes can remove small amounts of material for suitable costs only, they must not be applied to those parts: a conflict arises. On the other hand, conventionally forged parts may be milled by gear hobbing and gear hobbed parts may be grinded: a synergy exists. A different situation occurs in case of precision forged raw parts, which are characterized by small allowance and hardened, semi-rough surfaces. No specific synergy of conflict exists to the process combination hobbing/grinding, but specialized processes like hardfinishing exist for these parts. Knowing about the existence of these different manufacturing processes and their specific pre- and post-conditions, an order agent is able to select appropriate process combinations out of the available capabilities of the resources.

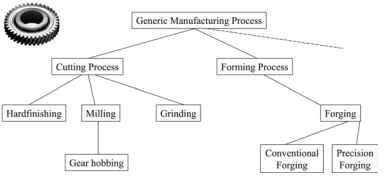


Fig. 4. Ontology of Manufacturing Processes

4.2 Use Case B: How to Make a Gear Wheel

The last paragraph illustrated how ontologies can be used to identify suitable sequence of available manufacturing processes. This identification depends on the propagation of available capabilities from the resource agent to the order agent and thus the ability of the resource agent to determine, if it possesses capabilities to perform a requested manufacturing task. Manufacturing tasks can be represented by manufacturing features (comparable to "design features" used as an object-oriented representation of product-related information like geometry in 3D-CAD systems). In addition, the resource agent may have knowledge to decide, if a manufacturing feature can be processed by a specific capability or not. If applicable, the requesting order agent is notified about specific capability. In this example, features e.g. for assembly operations of multiple parts into the whole product are not considered.

5 Conclusions and Future Work

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 than using 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 are 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. [4], [15], [5]. We are going to integrate approaches of approximate reasoning, as applied to adaptation of communication vocabularies [11].

The use cases are showing high potential for application of "weak" mapping of cabalibities to problems in the domain of distributed systems in cooperative manufacturing settings. Future work of us is focusing on the design and implementation of realistic application scenarios for capability management. Of course, more research and sophisticated algorithms for conflict assessment but also for building equivalence relations within the task agents are needed.

Acknowledgement. The authors would like to thank the reviewers for their valuable and helpful remarks. 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. Conen, W., and Neumann, G. *Coordination Technology for Collaborative Applications*. No. 1364 in Lecture Notes in Artificial Intelligence. Springer, Berling, 1998.
2. Durfee, E. H., and Rosenschein, J. S. Distributed problem solving and multiagent systems: Comparisons and examples. In *Proceedings of the 13th International Distributed Artificial Intelligence Workshop* (1994), pp. 94–104.

3. Marek, V., 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, September 2002.
4. Noy, N. F., and Musen, M. A. 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).
5. Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. Semantic matching of web services capabilities. In *Proceedings of the 1st International Semantic Web Conference (ISWC2002)* (2002).
6. Poslad, S., and Charlton, P. Standardizing agent interoperability: The FIPA approach. In *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* (New York, NY, USA, 2001), M. Luck, Ed., Springer-Verlag Inc., pp. 98–117.
7. Rao, A. S., and Georgeff, M. P. *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR & R-91).* Morgan Kaufmann Publishers, San Mateo, CA, 1991, ch. Modeling Rational Agents in a BDI-architecture, pp. 473–484.
8. Rao, A. S., and Georgeff, M. P. BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems* (San Francisco, 1995).
9. Smith, R. G. The contract net protocol: High-level communication and control in a distributed problem solver. In *Readings in Distributed Artificial Intelligence*, A. H. Bond and L. Gasser, Eds. Morgan Kaufmann, San Mateo, 1988.
10. Smith, R. G., and Davis, R. Frameworks or cooperation in distributed problem solving. *IEEE Transactions on Systems, man and Cybernetics 11*, 1 (1980).
11. Stuckenschmidt, H., and Timm, I. J. Adapting communication vocabularies using shared ontologies. In *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 2002), S. Cranefield, Ed., pp. 6–12.
12. Timm, I. J. Enterprise agents solving problems: The cobac-approach. In *"Informatik 2001" – Tagungsband der GI/OCG Jahrestagung, 25.-28. September 2001* (Universitaet Wien, 2001), K. Bauknecht, W. Brauer, and T. Mueck, Eds., pp. 952–958.
13. Toenshoff, H. K., Woelk, P.-O., Timm, I. J., and Herzog, O. Planning and production control using co-operative agent systems. In *Proceedings of the International Conference on Competitive Manufacturing (COMA '01)* (Stellenbosch, South Africa, 31st January — 2nd February 2001), D. Dimitrov and N. du Preez, Eds., pp. 442–449.
14. van de Vliert, E. *Complex Interpersonal Conflict Behaviour – Theoretical Frontiers.* Essays in Social Psychology. Psychology Press, East Sussex, UK, 1997.
15. Visser, U., Stuckenschmidt, H., Schlieder, C., Wache, H., and Timm, I. Terminology integration for the management of distributed information resources. *Kuenstliche Intelligenz 16*, 1 (2002), 31–34.
16. Wooldridge, M., and Lomuscio, A. *Proceedings of the Seventh European Workshop on Logics in Artificial Intelligence (JELIAI-2000).* Springer-Verlag, Berlin, 2000, ch. Multi-Agent VSK Logic.
17. Wooldridge, M. J. *Reasoning about Rational Agents.* The MIT Press, Cambridge, Massachusetts, 2000.