How to Build a Multi-Multi-Agent System: The Agent.Enterprise Approach.

Conference Paper · January 2004

Source: DBLP

CITATIONS
10

READS
641

4 authors, including:

Jens Nimis
Karlsruhe University of Applied Sciences
63 PUBLICATIONS **1,005** CITATIONS
SEE PROFILE

Thorsten Scholz
Universität Bremen
21 PUBLICATIONS **161** CITATIONS
SEE PROFILE

M. Stehli
GlobalFoundries Inc.
19 PUBLICATIONS **177** CITATIONS
SEE PROFILE

HOW TO BUILD A MULTI-MULTI-AGENT SYSTEM
The *Agent.Enterprise* Approach

Tim Stockheim
Institute of Information Systems, Universität Frankfurt,
Mertonstr. 17, D-60325 Frankfurt am Main, Germany
stockheim@wiwi.uni-frankfurt.de

Jens Nimis
Institute for Program Structures and Data Organization, Universität Karlsruhe (TH),
Am Fasanengarten 5, D-76131 Karlsruhe, Germany
nimis@ipd.uni-karlsruhe.de

Thorsten Scholz
Center for Computing Technologies (TZI), Universität Bremen,
Postfach 33 04 40, D-28334 Bremen, Germany
scholz@tzi.de

Marcel Stehli
Institute of Information Systems, Technische Universität Ilmenau,
Postfach 100 565, D-98684 Ilmenau, Germany
marcel.stehli@tu-ilmenau.de

Keywords: Agent-oriented software engineering, Multi-agent systems, Application integration

Abstract: The maturity of technical foundations for multi-agent systems and the support by development tools, infrastructure services, and a number of development methodologies leads to an increasing number of existing multi-agent systems. A more and more networked environment drives the demand for coupling these heterogeneous systems to large multi-multi-agent systems. Unfortunately, the design and implementation steps necessary in this context are currently not supported by established development methodologies; conventional approaches mainly focus on isolated multi-agent systems. In this paper, we present an approach for the integration of heterogeneous multi-agent systems. The *Agent.Enterprise* system is a coupled multi-multi-agent system that has been designed and tested in the manufacturing logistics domain.

1 INTRODUCTION

With the growing success of FIPA-standardization (http://www.fipa.org) for multi-agent systems (MAS) and the increasing availability of FIPA-compliant frameworks (e.g. JADE - http://jade.cselt.it) various MAS have been developed and real applications of MAS are emerging. These applications usually focus on a specific issue within a certain domain (e.g. manufacturing). In reality these systems often depend on each others' in- and output; therefore, the need for integration respectively coupling of these systems arises. Thus, coupled systems result in a large heterogeneous system. In this context we refer to the concept of coupled MAS as multi-multi-agent systems (MMAS). Each MAS represents a closed organizational entity with predetermined boundaries of its agents' influence. The MMAS restricts the communication between single MAS to prevent uncontrollable and unmanageable system complexity.

Nowadays, open service infrastructures such as Agentcities (http://www.agentcities.org) as well as technical foundations for the coupling of MAS are available. However, guidance in the software engineering process (analysis, design, implementation, testing) for these MMAS is missing, since existing agent-oriented development methodologies are focusing on isolated MAS only.

This paper presents an approach for analysis, design and implementation of MMAS that has been developed and applied by five research projects (Special interest group) for coupling their MAS. The resulting system is called *Agent.Enterprise* and was introduced in (Frey et al., 2003).

Outline. Section 2 considers and compares several existing MAS development methodologies. On this basis, we introduce the *Agent.Enterprise* approach as well as technical improvements (Gateway-Agent-Concept), which set up basic conditions for the approach in Section 3. The pros and cons of the proposed approach and conventional methodologies are discussed in Section 4. Finally, we conclude the paper in Section 5.

2 AGENT-ORIENTED DEVELOPMENT METHODS

Over the last years, the need for applicable and broadly accepted development methods for multi-agent systems resulted in a large number of efforts in order to overcome this problem. Various methods exist, which support at least one of the development phases (analysis, design, implementation, and deployment) with representations of varying formal accuracy and of varying semantic foundations, e.g. Gaia (Wooldridge, Jennings and Kinny, 2000), PASSI (Cossentino, and Potts, 2002), MASSIVE (Lind 2001), MaSE (Wood, and DeLoach, 2001), AUML (Odell, Parunak and Bauer, 2001).

The focus of most of the mentioned methods is on building single (most often closed) MAS and thus, none of them considers the development or integration of MMAS. Nevertheless, one can be expect, that the development process for MAS and MMAS will have some joint properties. The following subsections take a closer look at three of the most prominent methods in order to get a better understanding of the nature of MAS development methods before we present our method to interconnect different MAS.

2.1 GAIA

One of the best known representatives is the Gaia methodology for agent-oriented analysis and design (Wooldridge, Jennings and Kinny, 2000). It starts with a statement of requirements and ends up in a sufficiently detailed system that can be implemented directly. Hence, this methodology that consists of analysis and design steps can be thought of as a process of developing increasingly detailed models of the system to be constructed.

Analysis. The objective of the analysis stage is to gain an understanding of the system and its structure. The system's organization is viewed as a collection of roles that have certain relationships to each other, and that take part in systematic, institutionalized patterns of interaction with other roles. Thus, the organization model in Gaia comprises two further models: the role model and the interaction model.

The role model identifies the key roles in the system. Such roles are characterized by two types of attributes:

- The responsibilities of a role: A role captures a specific functionality. This functionality is represented by attributes known as the role's liveness and safety responsibilities.
- The permissions/rights associated with a role: A role has certain permissions, relating to the type and the amount of resources that can be exploited when carrying out the role.

The interaction model expresses relationships and dependencies between the various roles in a multi-agent system. Interactions need to be captured and represented in the analysis phase. Links between roles are represented by the interaction model. It consists of a set of protocol definitions, one for each type of inter-role interaction. A protocol is a pattern of interaction that has been formally defined and abstracted away from any particular sequence of execution steps.

Design. The aim in Gaia's design phase is to transform the analysis models into a sufficiently low level of abstraction so that traditional design techniques including object-oriented techniques may be applied in order to implement agents. The Gaia design process involves generating three models: The agent model identifies the agent types that will make up the system, and the agent instances that will be instantiated from these types. The service model identifies the main services that are required to carry out the agents' roles. Finally, the acquaintance model documents the communication links between the different agents.

2.2 PASSI

The Process for Agent Societies Specification and Implementation (PASSI) is a step-by-step requirement-to-code methodology (Cossentino, and Potts, 2002). It consists of five models and twelve process steps for building multi-agent systems and makes intense use of the UML notation.

Analysis. The analysis is described in detail by the system requirements model that focuses on the requirements in terms of agency and purpose. UML use-case diagrams are used to describe the agent domain and help to identify the agents and their roles within the system by analyzing their communication relationships.

Design. The design phase is characterized by the Agent Society Model that consists of an ontology description, a role description, and a protocol de-

scription. The ontology description visualizes two ontologies: the domain ontology for the involved entities and the communication ontology that models the agents' knowledge components and the inter-agent communication. The advantage of this approach is the ability to model the agents' knowledge and the communication specification as two related elements in the same way. The role description refines the defined agents' roles within the scope of the agents' lifecycles, collaboration, and communication. For formal communication protocol descriptions AUML is recommended.

Implementation. The agent implementation model defines the resulting system architecture by developing a multi-agent and a single-agent structure definition. Furthermore a behavior definition using activity diagrams is used to describe the multi-agent system.

All models have to be converted into a code model. Following the implementation, the deployment model is used to describe the location of the agents regarding the processing units and the migration and mobility constraints.

The strength of PASSI is the integration of known object oriented design methods into multi-agent system design.

2.3 MASSIVE

MASSIVE stands for "Multi-Agent SystemS Iterative View Engineering" and provides a framework for the development of multi-agent systems (Lind, 2001). It combines new software development approaches and standard software engineering techniques and applies them to multi-agent systems. The MASSIVE method differs from other traditional methodologies by being rather view-oriented than process-oriented, which makes it pointless to separate analysis and design.

Views are used to describe different aspects of the complete design. The logical decomposition of a system contains seven views: task, environment, role, interaction, society, architecture and system. The task view depicts the system from a functional perspective whereas the environment view analyses the accessibility of the environment from the system and the developer perspective. The role view describes the role model for the agents by analyzing the functional and physical relations within the system. The interaction view models the interaction necessary for solving a problem and the society view takes a macro perspective for building up structured collections of MAS entities. The resulting software model for the MAS and the single agents are described in the architectural and the system view.

The views are embedded in a stepwise refinement of the process model, the so called iterative view engineering. Due to this approach the model can deal with an incomplete problem specification and is not fixed for the entire project lifetime.

Another characterizing part of MASSIVE is the experience factory that provides a conceptual framework for enabling a systematic learning process within an organization. This way it is possible to improve the model according to the experience gained in the development process.

3 THE *AGENT.ENTERPRISE* APPROACH

The *Agent.Enterprise* initiative is a joint platform within the priority research program of the Deutsche Forschungsgemeinschaft to integrate recent research results and to join forces in order to build up a networked, agent-based application scenario for the manufacturing domain (Frey et al., 2003). In order to address the difficulties caused by the distributed structure of the involved projects, we developed a methodology based on aforementioned concepts of agent-oriented software engineering. The following subsections outline our scenario, the developed methodology, and some underlying design decisions.

3.1 SCENARIO

The research groups participating in *Agent.Enterprise* cover supply-chain-related aspects of enterprise co-operation as well as intra-organizational tasks of individual enterprises. In the context of the MMAS scheduling functions are offered by the DISPOWEB[*] project, shop floor production planning and control are provided by the KRASH[†] project, the IntaPS[‡] project and the FABMAS[§] project. Finally the ATT/SCC[**] project provides proactive tracking and tracing services to

[*] Dispositive Supply-Web-Coordination, Available from: http://www.dispoweb.de [Accessed 13.09.03]

[†] Karlsruhe Robust Agent Shell, Available from: http://www.ipd.uka.de/KRASH/ [Accessed 13.08.03]

[‡] Integrated Agent-based Process Planning and Production Control, Available from: http://www.intaps.org [Accessed 13.09.03]

[§] Agent-Based System for Production Control of Semiconductor Manufacturing Processes, Available from: http://www.tu-ilmenau.de/fabmas/ [Accessed 13.09.03]

[**] Agent-based Tracking and Tracing of Business Processes, Available from: http://www.wi2.uni-erlangen.de/research/ATT/index-e.html [Accessed 13.09.03]

guarantee the reliability of supply chain processes in the case of unforeseen disruptions.

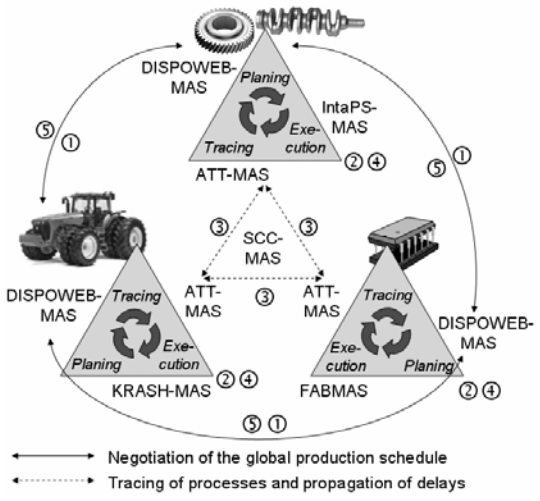

Fig. 1. Interaction in the integrated SCM architecture

A typical supply chain management cycle of distributed global planning of supply chain activities is shown in Figure 1. After generating an initial plan of orders and suborders comprising prices and dates of delivery, software agents located at the different supply chain partners carry out negotiations. Thereby, they optimize the cost and the due dates of deliveries (①).

These optimized delivery plans are used on the intra-organizational level inside each enterprise (i.e. in KRASH, IntaPS, and FABMAS) to plan the production of goods on each stage of the supply chain in detail. Three different MAS are concerned with varying aspects of production planning (②). They require the input from DISPOWEB agents and generate detailed plans for their production facilities.

These plans are the initial input for a controlling system, which is developed in the ATT/SCC project. The latter MAS monitors orders on every stage of the supply chain using a distributed architecture in order to proactively detect events that endanger the planned fulfillment. In case of such an event, e.g. a disruption in a production cycle, the ATT system informs the related partner enterprises about the event (③). This information can be used to trigger the rescheduling of the production steps on an enterprise level (④) or, in case of major events, even in the renegotiation of the contracts on the inter-enterprise level of the DISPOWEB system (⑤).

3.2 *AGENT.ENTERPRISE* METHODOLOGY

Unlike GAIA, PASSI, MASSIVE and other agent-oriented development methods the *Agent.Enterprise* methodology focuses on a distributed and weakly coupled development process, while minimizing the time required for face-to-face communication. Consequently, the initial design period is comparatively short and restricted to create the speech acts and interaction protocol design.

The result of the analysis and design process are consolidated in functionally restricted prototypes, which constitute a test bed for the components of the evolving MMAS. The projects substitute their prototypes with gateway-agents to connect their applications to the common scenario requiring a process of repeated cycles of redesign, implementation, and tests. Figure 2 depicts our development approach, while a detailed description can be found in the following subsections.

Analysis. To assign the participating projects to a specific functionality within the supply chain, the focus of the related research was taken as the major criterion which roles each project would play. Yet, the integration work starts solely with the Role Definition and Assignment followed by the Use Case Specification – the next step is to bring life to the roles.

A first approximation of the **Role Definition and Assignment** is made by performing a simple role-playing technique. To simulate the exchange of information between the systems a member of the scientific staff of each of the participating projects takes over the role of her MAS, and writes down its informational requirements. Then cards are handed out. The sender writes down the contents of the message as well as the receiver, therefore each card represents a single act of communication. Starting with the initiator of an order – the customer – the whole supply chain is acted out until finally the last card announces the delivery of the order from the OEM to the customer. As a result of this role-playing technique, the communication acts between the projects' MAS as well as the required information are specified informally and can be formalized into a **Use Case Specification** without requiring further interaction between the participating projects.

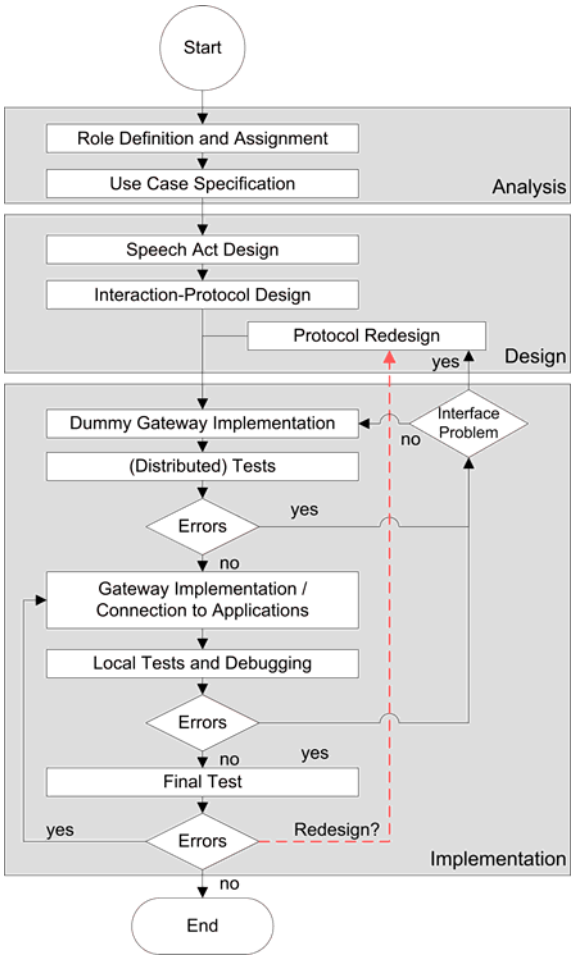

Fig. 2. The *Agent.Enterprise* development process

Speech Act Design. After defining the roles for each participating MAS, it is necessary to ensure that high-level communication between the systems is possible. Due to the heterogeneity of knowledge representation and semantics in the individual systems a language barrier for the communication exists. Consequently we introduce the so-called Gateway-Agent Concept, which is outlined in the next subsection. This concept defines a virtual MAS where the agents are scattered across a number of agent platforms such that an ontology can specify the semantics of conversations. While using ontological expressions as a means of communication, there are basically two methods to ensure that the partners understand each other: Shared ontologies or semantic mediation. As a first approach, we chose to agree on a shared ontology for communication between gateway-agents. In our scenario of 5 communicating gateways the setup cost for implementing semantic mediation would exceed the cost for agreeing on a shared ontology (Wache et al., 2001). Future work will include the semantic mediation based on a common terminology to further open the *Agent.Enterprise* system to other MAS.

The task of ontological modeling is performed using the method described in (Noy, and McGuinness, 2001), which is supported by an ontology modeling tool[††]. There is a number of tools, which support ontology-modeling, ranging from Protégé (Gennari et al., 2002), OilEd (Bechhofer et al., 2001) to WebOnto (Domingue, 1998). Protégé has proven to be best suited for the task of ontology modeling in our context. The Protégé-plug-in Beangenerator (van Aart et al., 2002) is able to generate code, which can be used for the communication funded on the modeled ontology. The code is deployed on the JADE agent platform, a development framework employed for several projects within the *Agent.Enterprise* group.

A starting point for an ontology modeling is to identify actions of the agents, which are requested from a communication partner. They are directly derived from assigned functionalities and specify tasks to perform. Afterwards, ontological concepts defining the artifacts to be dealt with in agent actions can be specified, e.g. the products to be manufactured. After modeling all the details for the supply chain, the concepts required for the supervision of all participating MAS are designed.

Interaction-Protocol Design. The next step in the overall design process is to define dynamics in conversations, i.e. which interaction-protocols have to be used for communication. The informal specification of interactions resulting from the card role-play is mapped to corresponding FIPA interaction protocols if available. As a result, the behavior of each gateway-agent for each MAS is specified as far as communication between gateways is concerned. The final step for integration is to ensure that the communication between the gateway-agent and the underlying MAS works.

(Distributed) Implementation. Based on the distributed structure of our research program, the development process takes into account comparatively long periods of independent development. Inspired by the concepts of Extreme Programming (Beck, 1999), the development process starts by the implementation of functionally restricted prototypes executing a simplified test case. These prototypes serve two purposes: On the one hand a consolidation of speech acts and interaction protocols is enforced. On the other hand, a test bed for autonomous develop-

[††] An overview of methodologies to develop ontologies has been given in Fernández-López, and Gómez-Pérez (2002).

ment emerges. The implementation of these prototypes is closely bound to test-sessions. The hereby gained experience is used for improvement and refinement of speech acts and/or interaction protocol design. The outcome of this work is a set of test modules and an exchange of experience within the covering research program. As an inevitable restriction of our approach, the simplification of the central projects' functions in the 'Dummy Gateway Implementation' could result in a setting where some aspects of the interaction protocols could not be sufficiently tested.

Subsequent to the completion of the prototypes, each project integrates its fully functional application into the test bed. Obviously each project sometimes has to debug prototypes of other projects. Resulting from clearly defined responsibilities for each prototype, explicit phases for consolidation are not intended.

3.3 THE GATEWAY-AGENT CONCEPT

Integration of complex systems needs agreements of technical nature in order to avoid a time-consuming struggle with implementation details. For the *Agent.Enterprise* approach two central design decisions are subsumed in the Gateway-Agent Concept, which is illustrated in Figure 3.

Firstly, the agreement upon the use of FIPA-compliant platforms avoids many of the communication-related obstacles and allows for concentrating on domain aspects. The second decision is that every individual MAS to be integrated should be represented by a single agent in the resulting MMAS.

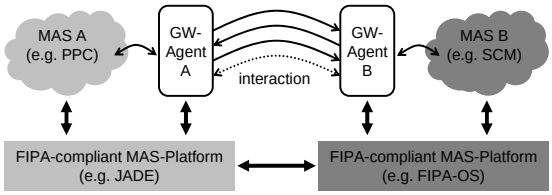

Fig. 3. The Gateway-Agent Concept

Thus, the interacting gateway-agents build up a virtual MAS. Together, these decisions can be seen as a specific MMAS architecture, which subsumes aspects of various well-known software patterns (Gamma et al., 1995):

- Façade pattern: The gateway-agents provide a unified interface to their MAS as a subsystem, comprising different roles and their respective functionality.
- Wrapper pattern (also called Adaptor): The gateway agents translate between internal formats and behavior of their corresponding MAS and the common representation in the virtual MAS.
- Bridge pattern: The different types of the gateway agents provide abstract interfaces decoupled from the implementation of their MAS. In the next section we give an example of three MAS that play the role of a supplier in a supply chain scenario. In the virtual MAS they all are represented by the same type of gateway agent while their implementation is completely different and independent.

There are several advantages of the Gateway-Agent Concept, e.g., developers can focus their effort on a single agent and only the gateway-agents must be available during operation. Also, there is no restriction in the centralization of the different roles of the MAS into a single gateway-agent, as the different functionalities of this agent can be redirected to several other agents in the MAS it represents.

4 BENEFITS OF THE *AGENT.ENTERPRISE* METHODOLOGY

In spite of the many differences of the presented methods, e.g. in respect to scope, representation format, development process, and semantic foundation, they share some central concepts. Agents provide functionality by taking over certain tasks associated with a role. They interact in order to build up societies and have a domain model of the environment in which they are situated.

The presence of the concepts and their corresponding models in the described development methods on one hand raises the question if and how they must be addressed by a design process, which aims at the integration of several existing independent MAS to an MMAS. On the other hand, such an integration method needs not to cover all aspects of agent-oriented development. It has to concentrate on the requirements from the integration of different already existing systems.

	Gaia	**PASSI**	**MASSIVE**	***Agent.Enterprise***
Functionality	- (input)	Domain Description	Task View	- (input)
Roles / Agents	Role Model / Agent Model	Agent and Roles Identification / Role Description	Role View	Role Identification and Description
Tasks	Role M. (activities), Services Model	Task Specification	Task View	- (input)
Interaction	Interactions Model / Services Model	Protocols Description	Interaction View	Protocols Description
Domain Knowledge	-	Ontology	-	Shared + Individual Ontologies
Macro Level	- (not yet)	(Role Description)	Society View	(Role Description)
Environment	Role M. (rights, perm. and responsibilities)	-	Environment View	-
Architecture / Implementation	-	Agent Implementation and Code M.	Architectural View	Gateway Model
Deployment	Acquaintance Model	Deployment Model	System View	(not yet)

Table 1. MAS-specific concepts and aspects covered by the discussed methods.

Although our *Agent.Enterprise* approach is comparable to fully-fledged development methods, it rather has to be seen as an addition to such methods. For example, the functionality of the resulting MMAS derives from existing MAS instead of being newly designed. Hence certain aspects of the different approaches can not be compared (cp. Table 1). In the following we emphasize three aspects where conventional methodologies could not be applied.

Architecture. The Gateway-Agent Concept calls for the assignment of a specific functionality to the corresponding gateway-agent. One could say that the gateway-agents give a macro level view of the MMAS as representatives of their specific MAS. The distinction between functionality and role can be unattended in this context, as a specific functionality is carried out by individual MAS and therefore its gateway-agent is required to play the role associated with the functionality.

Domain Knowledge. In most development methods, the modeling of the domain knowledge does not play a central role. Often an implicit common domain model is assumed when a (closed) MAS is developed from scratch. This assumption does not hold for the integration of existing MAS. Thus, the design of a domain model, which is common to all gate way-agents, and suitable at least for communication purposes becomes a major development step.

The same applies to the **interaction** design: it has to consider the dynamic aspects of the functionality provision and leads to a number of communication protocols. A basic experience of *Agent.Enterprise* is the necessity of a bottom-up approach for the design of interactions caused by the established functionality of the participating MAS. The interaction design comprises three elements which are similar to the PASSI method (but less formalized): informal specification, mapping to FIPA-interaction protocols (AUML), and ontology-based content specification (e.g. Protégé).

All other mentioned aspects such as accessibility of the environment and deployment can be left to the individual projects as no overall design guideline is necessary for integration. The result of the application of our methodology is the fully functional *Agent.Enterprise* MMAS.

5 CONCLUSION

This paper introduces a new methodology for coupling MAS using methods from agent-oriented design. The existence of sufficiently specified concepts applicable for most steps of our prospected methodology made little conceptual effort necessary.

Our approach addresses special needs, e.g. dealing with individual ontologies that arise from the nature of MAS integration. Special attention is given to agreements on the technical system architecture. The introduced technical concept, called Gateway-Agents, accelerates the implementation by enforcing those necessary agreements on technical standards. Furthermore we could replace an often inflexible standardization of interfaces by tool-supported verbal agreements. Moreover, the *Agent.Enterprise*

methodology is designed to fit the need for distributed development.

The resulting MMAS called *Agent.Enterprise* covers services in the range of supply chain scheduling, shop floor production planning and control, and proactive tracking and tracing services. By building this large-scale adaptable MMAS we showed the applicability of our methodology. The provided reliability of overall supply chain processes also supports our assumption that MMAS will turn out to be a valuable extension to MAS esp. in the context of Supply Chain Management.

6 ACKNOWLEDGEMENTS

This work as well as the participating projects is funded by the Deutsche Forschungsgemeinschaft (DFG) within the German priority research program 1083 "Intelligent Agents in Real-World Business Applications" (refer to http://www.realagents.org for further information).

We also like to thank Daniel Pfeifer, Michael H. Schwind and Ingo J. Timm for their contributions.

REFERENCES

van Aart, C. J., Pels, R.F., Giovanni, C. & Bergenti, F. (2002) 'Creating and Using Ontologies in Agent Communication', *Proc. of the Workshop on Ontologies in Agent Systems, 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS2002)*, Bologna, Italy.

Bechhofer, S., Horrocks, I., Goble, C. & Stevens, R. (2001) 'OilEd: a Reason-able Ontology Editor for the Semantic Web', *Proc. of Joint German/Austrian conference on Artificial Intelligence (KI2001)*, Springer-Verlag.

Bellifemine, F., Poggi, A. & Rimassa, G. (2001) 'JADE: a FIPA2000 Compliant Agent Development Environment', *Proc. of the Fifth International Conference on Autonomous Agents*, ACM Press.

Cossentino, M. & Potts, C. (2002) 'A CASE tool supported methodology for the design of multi-agent systems', *Proc. of the International Conference on Software Engineering Research and Practice*, CSREA Press.

Domingue, J. (1998) 'Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web', *Proc. of the 11th Workshop on Knowledge Acquisition for Knowledge-based Systems*, Banff, Canada.

Fernández-López, M. & Gómez-Pérez, A. (2002) 'Overview and Analysis of methodologies for building ontologies' The Knowledge Engineering Review (KER), vol. 17 [2]. pp: 129-156.

Frey, D., Stockheim, T., Woelk, P.-O. & Zimmermann, R. (2003) 'Integrated Multi-agent-based Supply Chain Management', *Proc. of the 1st International Workshop on Agent-based Computing for Enterprise Collaboration*, IEEE Computer Society Press.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995) '*Design Patterns: Elements of Reusable Object-Oriented Software'*, Addison-Wesley, Reading, Mass.

Gennari, J., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., Noy, N. F. & Tu, S. W. (2002) 'The Evolution of Protégé: An Environment for Knowledge-Based Systems Development', Medical Informatics Technical Report SMI-2002-0943, Stanford.

Lind, J. (2001) 'Iterative Software Engineering for Multiagent Systems: the MASSIVE Method', *Lecture Notes in Computer Science and Lecture Notes in Artificial Intelligence*, Vol. 1994, Springer-Verlag.

Noy, N.F. & McGuinness, D.L. (2001) 'Ontology Development 101: A Guide to Creating Your First Ontology'. Stanford Knowledge Systems Laboratory Tech. Report KSL-01-05 and Stanford Medical Informatics Tech. Report SMI-2001-0880, Stanford, CA.

Odell, J., Van Dyke Parunak, H. & Bauer, B. (2001) 'Representing Agent Interaction Protocols in UML', *Proc. of the First International Workshop on Agent-oriented Software Engineering*, Springer-Verlag.

Special Interest Group on Manufacturing Logistics of the German priority research program 1083 "Intelligent Agents in Real-World Business Applications"

Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann N. & Hübner, S. (2001) 'Ontology-Based Integration of Information - A Survey of Existing Approaches', Gómez-Pérez, A., Gruninger, M., Stuckenschmidt, S. & Uschold, M. (eds.), *Proc. of the IJCAI-Workshop Ontologies and Information Sharing*, Seattle, WA, pp. 108-117.

Wood, M.F. & DeLoach, S.A. (2001) 'An Overview of the Multiagent System Engineering Methodology', *Proc. of the First International Workshop on Agent-oriented Software Engineering*, Springer-Verlag.

Wooldridge, M., Jennings, N.R. & Kinny, D. (2000) 'The Gaia Methodology for Agent-Oriented Analysis and Design', *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, pp. 285–312.