Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

An Automation Agent Architecture with A Reflective World Model in Manufacturing Systems

Mathieu Vallée*, Hermann Kaindl*, Munir Merdan**, Wilfried Lepuschitz**, Edin Arnautovic* and Pavel Vrba‡
*Institute of Computer Technology, Vienna University of Technology, Vienna, Austria
Email: {vallee, kaindl, arnautovic}@ict.tuwien.ac.at
**Automation and Control Institute, Vienna University of Technology, Vienna, Austria
Email: {merdan, lepuschitz}@acin.tuwien.ac.at
‡Rockwell Automation Research Center, Pekarska 10a, Prague 155 00, Czech Republic
E-mail: pvrba@ra.rockwell.com

***Abstract*—Manufacturing systems have become very complex, and the traditional hierarchical and centralized approaches are not adequate any more. Decentralized approaches are considered promising, but they are not yet sufficiently understood for widespread industrial application. In particular, agent-based control has not yet achieved its potential, so that more research is still required.**

Therefore, we propose a new architecture of *automation agents*. Such an agent is composed of a hardware component and a software component, where the hardware component can be viewed as an embodiment of this software component in its manufacturing environment. An important part of the software component is a world model repository of the automation agent. As a special innovation, the world model is *reflective* in the sense, that it contains a symbolic representation of the automation agent and of its relations to its environment as well. The world model repository is made up of representations of situations and activities. In summary, we propose an automation agent architecture with a reflective world model for use in manufacturing systems.

***Index Terms*—Manufacturing Systems & Automation, Distributed Intelligent Systems, Agent Architecture, World Model**

I. Introduction

A manufacturing system, which is made of subsystems that could be complex systems themselves, is a typical example of a complex environment. It is influenced by extremely turbulent conditions and characterized with a high number of system states. Such systems are becoming even larger and more complex, due to the current trends such as mass customization, lot size one production, company spanning supply chains, and supplier networks. The complexity is manifested not only in the systems themselves, but also in the processes, in the products to be manufactured, and in the company structures.

Traditional hierarchical and centralized approaches are not adequate for the control of such distributed systems and can fail due to insufficient capabilities to cope with the high degree of complexity and practical requirements for robustness and flexibility. The lack of flexibility and adaptability of such systems leads to deviations from the initial working plans. Such situations occur when certain resources become unavailable or additional resources are introduced into these systems. These deviations cause delays and non-operative conditions that increase costs. Poor robustness and resource utilization are also weaknesses of centralized systems because all control functionality is concentrated in a single point.

The application of decentralized control architectures, based on autonomous and co-operative entities (i.e., holons, agents, fractals or cells), is considered as a promising approach for handling the system complexity and dynamics [1]. Intelligent agents offer a convenient way of modeling processes that are distributed over space and time, making the control of the system decentralized [2]. This increases flexibility and enhances fault tolerance [3]. This approach replaces a centralized database and control computer with a network of agents, each endowed with a representation of its local environment and the ability and authority to respond locally to that environment. So, the ability to maintain an accurate internal representation of pertinent information about the environment in which it operates, is seen as a major challenge in autonomous systems to be solved in order to overcome the issues mentioned above.

We address this challenge in our paper and present an architecture that includes a world model repository. This repository is part of a software component that is assigned to a hardware component. The latter can be understood as an embodiment of this software component. Taken all this together, we create an *automation agent*.

The world model of an automation agent contains a symbolic representation of the manufacturing domain, more precisely in terms of situations and activities. For this representation, we make use of ontologies.

A key issue is the representation that an automation agent has about itself. It is clear that it will need to maintain certain attributes, e.g., about its physical position. Our approach is more ambitious in this respect, since we represent the automation agent itself in the same symbolic way as its environment. In addition, this symbolic representation contains the relationships of this automation agent with other entities in its environment. In this sense, its own word model contains a representation of its *self*. And in this sense, we call this world model *reflective*.

For explaining all this, we use the following running example. We choose a conveyor, which is a representative part of a manufacturing system. Its main function is to move the belt and transport the material on the belt forward. We assign a software component to a conveyor and create a conveyor automation agent in this way. We also sketch how such an automation agent would act under failure conditions.

The remainder of this paper is organized in the following manner. First, we review the state of the art in terms of related work. Then we present our concept of an automation agent and its architecture. Based on that, we elaborate on its world model repository with a focus on reflection first, and finally describe its essential content.

II. State of the Art

Substantial research has been devoted to apply the agent paradigm in the manufacturing control domain. Bussmann and Schild presented the deployment of agent-based control for the management of a flexible transportation system [4]. Series of simulations showed that the agent-based control is extremely robust against disturbances of machines as well as failures of control units. The control system has been installed in the DaimlerChrysler plant in Stuttgart validating the simulation results in a real manufacturing environment. Maturana et al. reported the successful implementation of a multi-agent system (MAS) for the distributed control of a ship equipment with the aim to reduce the manning and to improve failure tolerance of US Navy shipboard systems [5]. The developed MAS architecture has been tested both in a simulation environment and on the Navy's Reduced Scale Advanced Demonstrator model featuring 116 agents running in 22 Logix controllers. The applied architecture demonstrated system scalability, reconfiguration ability as well as a high failure tolerance level [6]. Vrba et al. successfully deployed the Manufacturing Agent Simulation Tool (MAST) on the manufacturing testbed at the Automation and Control Institute (ACIN), Vienna University of Technology [7]. The MAST system is used to control the physical palette transfer system and its functionality was tested in diverse failure scenarios (e.g., conveyor failure). The presented agent control system verified its ability to effectively handle disturbances in the real manufacturing environment.

Nevertheless, although confirmed as a promising approach and successfully deployed in a number of different applications, the wide adoption of agent-based concepts by the industry is still missing. Interestingly, relevant theoretical work on agent architecture [8], multi-agent system coordination [9], as well as methodologies for designing multi-agent systems [10], is rarely considered in implemented prototypes. Besides, currently used agent platforms do not always fulfill the requirements of practical automation applications concerning the real-time constrains and resource capabilities [11].

Lack of awareness of the potentials of agent technology [12], missing trust in the idea of delegating tasks to autonomous agents [13] and a "pioneer" risk that accompanies every new technology [14] are recognized as the main reasons for the lack of real industrial applications. Besides, there are serious concerns regarding the stability, scalability and survivability, especially in unpredictable environments of attacks and system failures [15]. In particular, the architectural design of both individual agents and multi-agent systems is critical to cope with the increasing size and complexity of these systems. Relevant solutions should give a better understanding of the agents' behavior, as well as how they obtain and represent knowledge about the world [16].

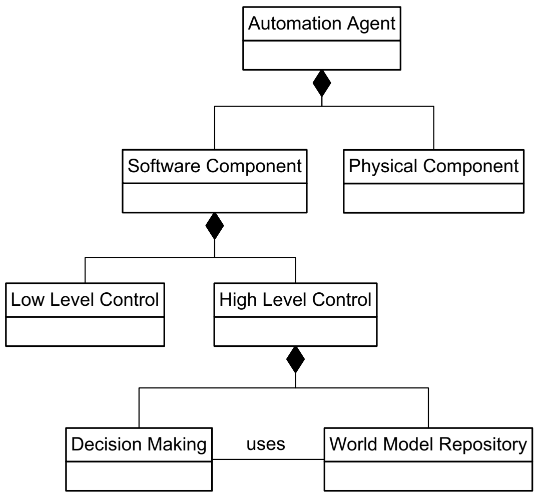

Fig. 1. Composition of the automation agent.

The integration of the agent-based system with real-time information and the IEC 61499 function blocks technology is seen as a promising approach to improve its efficiency and effectiveness [17]. Besides, transformation of received and perceived raw data into knowledge and effective reasoning on this data is considered as an important issue to be investigated [18]. Moreover, the development of tools, techniques and methodologies that could ensure easier and more abstract ways of agent system development, modification and management could lead to a higher rate of acceptance as well as better understanding of the agent technologies.

III. Automation Agent

We now present more precisely the architecture of an automation agent. While Sünder et al. [19] defined the notion of automation component as the basic unit of a hierarchical control system, we go one step further and define the notion of automation agent as the basic unit for a *distributed intelligent control system*. In such a system, control is decentralized: automation agents should be capable of directly coordinating themselves in order to accomplish complex tasks. In addition, each automation agent should provide a certain level of adaptation and fault-tolerance at its own level.

Fig. 1 shows a UML[1] representation of the composition of an automation agent. An automation agent is composed of a physical component and a software component. The physical component can be viewed as the *embodiment* of the automation agent. In a manufacturing system, it is typically a mechatronic

[1]At the time of this writing, the specification of UML is available at `http://www.omg.org`.

component consisting of both physical hardware and interface to the software part. The software component serves as an "artificial intelligence" of the physical component. It controls the physical component and it is supposed to guide it to achieve the goals of the automation agent.

The software part can be further decomposed to distinguish between low-level control (LLC) and high-level control (HLC) [20]. More precisely, the LLC and HLC are organized in a layered architecture, where HLC is on top of the LLC. The HLC and LLC layers communicate over a defined interface, as shown in [21]. The LLC layer is responsible for directly controlling the physical component, using a limited set of reactive behaviors. The HLC layer is responsible for the control of the global behavior of the agent, in order to ensure both the achievement of its own goals and the coordination with other automation agents of the system. A fundamental distinction between the layers is the ability of the LLC behaviors to execute in real-time, while the HLC behaviors may require longer computations without timing guarantees, interactions with other agents or even interactions with human operators. This architecture belongs to the class of layered, hybrid architectures [8], [22].

Finally, we introduce a decomposition of the HLC to distinguish between the world model repository and a decision-making component. The world model repository contains a symbolic representation of the world of the agent (both situation and activities). It is also equipped with mechanisms for detecting states the agent should be aware of. The decision-making component is in charge of reasoning about these states and deciding what to do (e.g., communicate with other agents, request an action from the LLC).

This architecture is applied to the automation agent that we use as a running example. Its physical component is a conveyor, i.e., a mechatronic component composed of a conveyor belt, a motor, a power relais and inductive sensors for sensing the presence of pallets. Its software component is then involved in controlling the speed of the conveyor, monitoring its load and diagnosing failures. Within the software component, the LLC is responsible for monitoring when pallets enter and leave the conveyor, using the inductive sensors. It is also capable of detecting failures of the motor, which prevent the normal operation of the conveyor. Based on the information provided by the LLC, the HLC is capable of more complex control. For instance, its world model repository contains knowledge about pallets entering and leaving the conveyor, and it can detect an anomaly when a pallet entered the conveyor but did not leave it at the expected time, although no motor failure was detected. When such an anomaly appears, its decision-making component interacts with other agents to warn them about the problem, and possibly to solve it (e.g., another agent may have detected the missing pallet, which could indicate a sensor failure).

IV. World Model Repository

We now focus on the world model repository, which plays a central role in our architecture. Especially, we first highlight how this world model repository is built using a certain kind of reflection, based on the symbolic representation of the automation agent itself. We then detail the content of the world model, and how it is used in our running example.

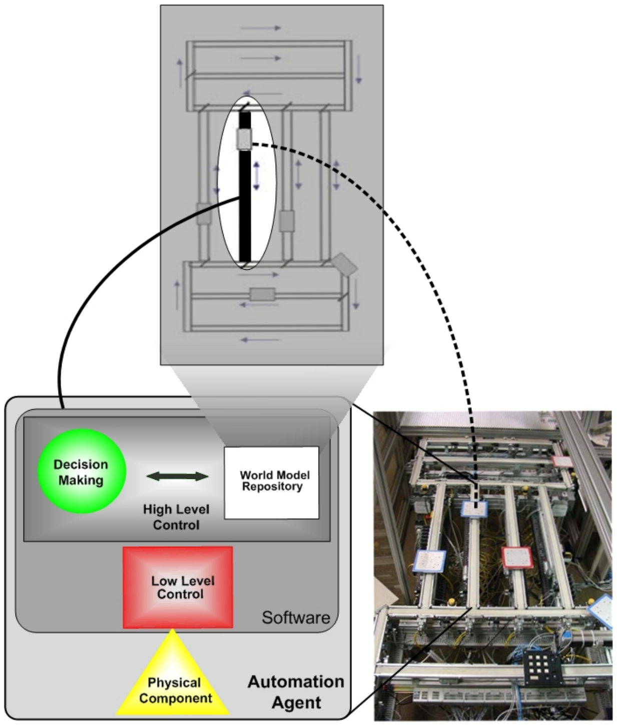

Fig. 2. Schematic view of the agent architecture with the world model repository.

A. Reflection

Especially when an agent is an integral part of its environment, we argue in favor of integrating all the information that the software has represented of the agent itself and of its relations to other objects into its model of the domain, resulting in a *reflective* world model.

Fig. 2 illustrates this idea. Its right part contains a photograph of a test-bed of a manufacturing and transport facility, including several physical components. The top part shows a schematic view of a model of this facility. The dotted line between the photograph and the schematic model shows the correspondence of one such physical component with its representation in the model. The schematic view of the automation agent left of the photograph zooms into such a physical component. The world model repository of this agent contains the model of the facility illustrated in the top part of Fig. 2.

The key point is that this symbolic model includes a representation of this very component that contains this model (shown with the full line). That is why we call it a *reflective* model. Such a reflective model makes the software architecture reflective in a certain sense. We focus on relations of the

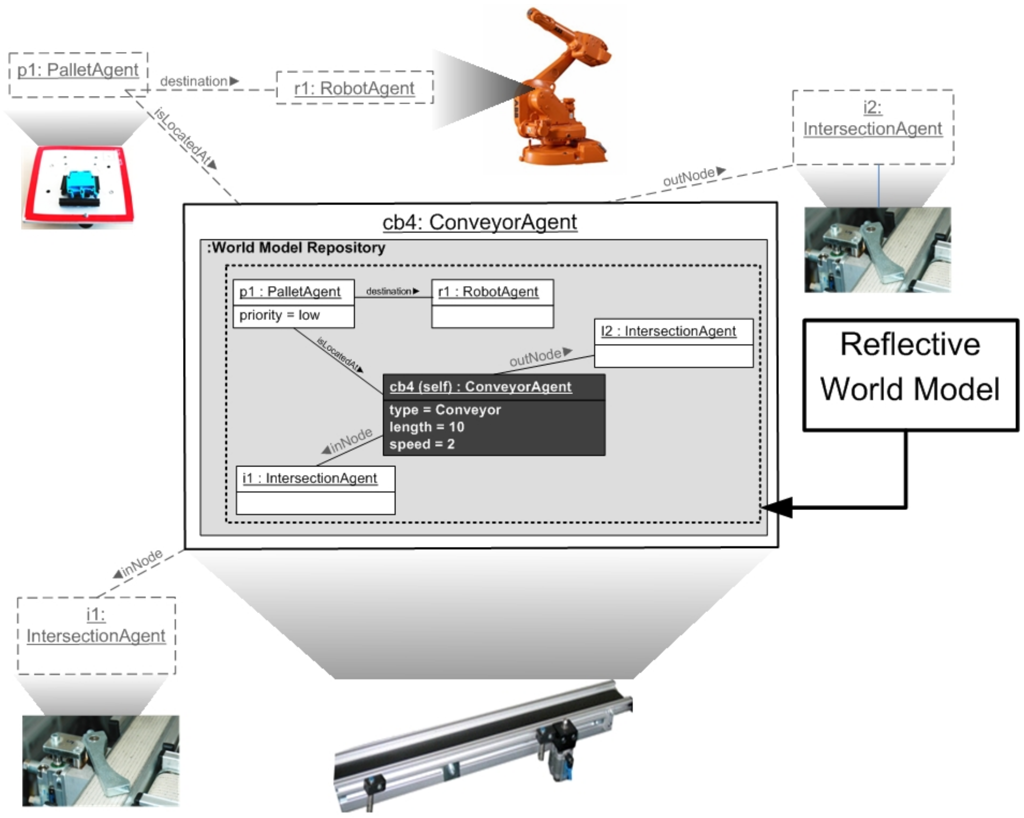

Fig. 3. Reflective world model with instances.

agent to other agents in the domain, attributes and operations through which it can have influence on its environment. So, this is an *external* view of the agent as represented within this architecture. Note that this external view is in contrast to the widely published view of reflection where software has representations of its *internal* construction.

Fig. 3 shows the reflection in terms of object instances. The boxes at the fringe represent the environment. This environment contains instances of several types of automation agents: $r1$:RobotAgent, $i2$:IntersectionAgent, etc. Some of these instances contain relations to each other (dashed lines), e.g., relation *isLocated* between the PalletAgent $p1$ and the ConveyorAgent $cb4$. Agent $cb4$ of the type ConveyorAgent is thus part of this environment and shown in the middle of the figure. The figure is (partly) also a white-box view of this very agent (instance), since it shows also the world model inside its component named WorldModelRepository, which would ideally be the same model as the one that we see here of the environment. Most importantly, this world model inside the software of $cb4$ contains a representation of $cb4$ itself. This figure, however, may suggest that there is infinite regress. This can be easily avoided when the box of itself in its world model is viewed and handled as a representation of a kind of *self*, which is treated as a black box and not zoomed in further.

Based on the generic world representation, an agent will create its own world model through the instantiation of these generic concepts. The information needed for that will be retrieved through sensors and communication with other agents. It is important to note that the agent is aware of the *self* within this model and, therefore, has a “clean” representation of its own relations to other entities in the environment. It will also “fill” its own current properties in this reflective world model. The reflective agent will also have a representation of its own abilities in terms of activities which it can perform.

B. Content of the World Model

We now give more details about the content of such a reflective world model, and we show how it is used in our running example. Fig. 4 depicts the internal structure of the world model. It consists of two parts:

- The *situation model* contains knowledge about the agent situation. The situation of an agent consists both of its own characteristics (e.g., the length of the conveyor) and its relations to other entities in the world (e.g., the intersection from which pallets enter the conveyor). The situation model does not only contain a representation of *facts* describing the situation, but also the *ontologies* to which the representation of facts conforms. Ontologies

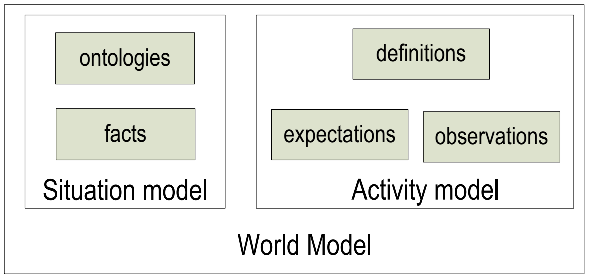

Fig. 4. Content of the world model.

express more generic knowledge about how the world is represented, while facts express knowledge about the current state of the world.

- The *activity model* contains knowledge about activities of the agent. It expresses both knowledge about *observations* (i.e., instances of activities that take place and are observed by the agent) and more abstract knowledge about possible activities. The latter does not refer to instances of activities, but rather to categories or sets of activities. Within this more abstract knowledge, we further distinguish between *definitions* (i.e., knowledge which defines the activities the agent can be involved in) and *expectations* (i.e., knowledge about the activities the agent effectively expects to be involved in).

These two models offer a complementary view of the world of the agent. Especially, the situation model offers a rather static, but more intuitive, view of the entities present in the world and how they are related to the agent itself. The activity model expresses more precise knowledge, especially involving time and agent roles in the activities, as well as dependencies between activities and generalization/specialization relationships between activities. The separation between these two models facilitates the definition of the agent knowledge, as the content of the situation model is based on ontologies which can be easily understood by domain experts. The content of the activity model can be automatically generated from the situation model based on an ontology and rules. Moreover, the representation of *self* is present in both models and serves as a pivot for managing interaction between them.

In the context of our running example, we now detail how an automation agent equipped with a reflective world model is able to detect an anomaly when a pallet failed to leave the conveyor as expected. To illustrate this example, we consider a scenario involving several steps of operation of the automation agent. Fig. 3 already gives an illustration of the content of the situation model resulting from this scenario. To further illustrate this example, and especially the role of the activity model, we now detail the successive steps of the scenario:

1) First of all, the automation agent is assigned a name, for instance $cb4$. When the software starts up, an explicit association between this name and *self* is created in the situation model. In Fig. 3, this representation of *self* corresponds to the black box in the middle, without any content yet. From now on, the agent will be able to detect that any mention of $cb4$ refers to itself.
2) After startup, the automation agent attempts at getting information about its own configuration. In our case, it is provided with the information that $cb4$ (i.e., itself) is a “ConveyorAgent”, whose length is 10 meters and speed is 2 meters/second. The corresponding symbolic knowledge is added to the situation model. In Fig. 3, this corresponds to the content of the black box representing the agent itself.
3) From the knowledge that is now present in the situation model, it is possible to deduce some definitions of activities. In particular, because it is a “ConveyorAgent”, $cb4$ is capable of transporting pallets. This is represented by adding the definition of a category of activities called “TransportPallet”. Because of the length and speed, any instance of this category must have a duration of 5 seconds.
4) As part of the configuration of the system, the automation agent also receives information from other agents. Here, $cb4$ is informed that an agent named $i1$ serves as its input node and an agent called $i2$ serves as its output node. The corresponding symbolic knowledge is added to the situation model. This is illustrated in Fig. 3 by the relations *inNode* and *outNode* between the representation of the agent $cb4$ itself and the representations of the agents $i1$ and $i2$ (respectively).
5) From this new knowledge in the situation model, the definitions contained in the activity model are refined. A new category of activities called “TransportPalletBetweenI1andI2” is introduced as a specialization of the previous category “TransportPallet”. Additionally, the agent should be aware that, when a “TransportPalletBetweenI1andI2” activity is performed, two sub-activities should also be performed: detecting that a pallet comes from $i1$ and detecting that a pallet leaves to $i2$. This leads to the introduction of two corresponding categories of activities called “DetectPalletFromI1” and “DetectPalletToI2”. These two categories are related to the “TransportPalletBetweenI1andI2”, as “DetectPalletFromI1” should be performed at the beginning of “TransportPalletBetweenI1andI2” and “DetectPalletToI2” should be performed at the end of “TransportPalletBetweenI1andI2”.
6) During the operation of the system, the detection that a particular pallet $p1$ leaves the input node $i1$ at a particular time $t = 15$ leads to two modifications of the world model. First, the situation model is updated to indicate that $p1$ is now located at $cb4$. This is illustrated in Fig. 3 by a relation *isLocatedAt* between the representation of the agent $cb4$ itself and the representation of $p1$. Second, the activity model is updated to represent the observation of an activity of type “DetectPalletFromI1”, which started at time 15.
7) After the addition of the new observation, the activity model is further updated to represent the expectations that derive from this observation. First, the observation

of an activity "DetectPalletFromI1" at time 15 leads to adding the expectation for an activity "Transport-PalletBetweenI1andI2" starting at time 15 and ending at time 20 (because its duration should be 5). Second, the addition of this expectation leads to adding another expectation for an activity "DetectPalletToI2" at time 20.

8) If we now consider the case that the pallet $p1$ got stuck on the conveyor (which is not possible to detect directly using a sensor), the expectation for an activity "DetectPalletToI2" at time 20 is not confirmed by a corresponding observation. In such a case, an inconsistency is detected between the reflective world model and the observations of the world, indicating a possible anomaly.

V. Conclusion

In this paper, we propose the novel concept of an *automation agent* together with its innovative architecture. Such an automation agent is composed of a hardware component and a software component assigned to it. The interesting part of this software component is its high-level control, which is composed of software for decision making and a world model repository. This repository contains knowledge, information and data about situations and activities. Its really innovative feature is that this world model contains a representation of the *self* of this very automation agent. This makes the world model *reflective*.

The implementation of this concept is already under way at the time of this writing. We have a laboratory of manufacturing systems in place, where this new automation agent will be integrated and tested. We also plan to make thorough experiments especially designed for the reflective world model.

Acknowledgement

The authors would like to acknowledge the financial support from the FIT-IT: Semantic Systems program, an initiative of the Austrian federal ministry of transport, innovation and technology (bm:vit) under contract FFG 815132.

References

[1] A. Tharumarajah, A. Wells, and L. Nemes, "Comparison of emerging manufacturing concepts," in *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, 1998, pp. 325–331. [Online]. Available: http://dx.doi.org/10.1109/ICSMC.1998.725430

[2] N. Jennings and S. Bussmann, "Agent-based control systems: Why are they suited to engineering complex systems?" *IEEE Control Systems Magazine*, vol. 23, no. 3, pp. 61–73, 2003. [Online]. Available: http://dx.doi.org/10.1109/MCS.2003.1200249

[3] P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, available online 17 November 2008. [Online]. Available: http://dx.doi.org/10.1016/j.engappai.2008.09.005

[4] S. Bussmann and K. Schild, "An agent-based approach to the control of flexible production systems," in *Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation*, vol. 2, 2001, pp. 481–488. [Online]. Available: http://dx.doi.org/10.1109/ETFA.2001.997722

[5] F. P. Maturana, P. Tichý, P. Šlechta, F. Discenzo, R. J. Staron, and K. Hall, "Distributed multi-agent architecture for automation systems," *Expert Systems with Applications*, vol. 26, no. 1, pp. 49–56, 2004. [Online]. Available: http://dx.doi.org/10.1016/S0957-4174(03)00068-X

[6] P. Tichý, P. Šlechta, R. Staron, F. Maturana, and K. Hall, "Multiagent technology for fault tolerance and flexible control," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 36, no. 5, pp. 700–704, 2006. [Online]. Available: http://dx.doi.org/10.1109/TSMCC.2006.879381

[7] P. Vrba, V. Marík, and M. Merdan, "Physical deployment of agent-based industrial control solutions: Mast story." in *Proceedings of the IEEE International Conference on Distributed Human-Machine Systems*, Athens, Greece, 2008, pp. 133–139.

[8] J. P. Müller, *The Design of Intelligent Agents: A Layered Approach*, ser. LNAI. Springer, 1996, vol. 1177. [Online]. Available: http://dx.doi.org/10.1007/BFb0017806

[9] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. N. Prasad, A. Raja, R. Vincent, P. Xuan, and X. Q. Zhang, "Evolution of the GPGP/TÆMS Domain-Independent coordination framework," *Autonomous Agents and Multi-Agent Systems*, vol. 9, no. 1, pp. 87–143, July 2004. [Online]. Available: http://dx.doi.org/10.1023/B:AGNT.0000019690.28073.04

[10] L. Padgham and M. Winikoff, *Agent-Oriented Software Engineering III*. Springer, 2003, ch. Prometheus: A Methodology for Developing Intelligent Agents, pp. 174–185. [Online]. Available: http://dx.doi.org/10.1007/3-540-36540-0_14

[11] S. Theiss, V. Vasyutynskyy, and K. Kabitzsch, "Software agents in industry: A customized framework in theory and praxis," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 2, pp. 147–156, 2009.

[12] M. Pěchouček, M. Rehák, and V. Marík, "Expectations and deployment of agent technology in manufacturing and defence: case studies," in *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*. The Netherlands: ACM, 2005, pp. 100–106. [Online]. Available: http://dx.doi.org/10.1145/1082473.1082811

[13] K. Sycara, "Multiagent systems," *AI Magazine*, vol. 19, no. 2, pp. 79–92, 1998. [Online]. Available: http://www.aaai.org/AITopics/assets/PDF/AIMag19-02-2-article.pdf

[14] M. Pěchouček and V. Marik, "Industrial deployment of multi-agent technologies: review and selected case studies," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 3, pp. 397–431, Dec. 2008. [Online]. Available: http://dx.doi.org/10.1007/s10458-008-9050-0

[15] A. Helsinger, M. Thome, and T. Wright, "Cougaar: a scalable, distributed multi-agent architecture," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, 2004, pp. 1910–1917 vol.2. [Online]. Available: http://dx.doi.org/10.1109/ICSMC.2004.1399959

[16] J. Lastra and M. Delamer, "Semantic web services in factory automation: fundamental insights and research roadmap," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 1, pp. 1–11, 2006.

[17] W. Shen, L. Wang, and Q. Hao, "Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 36, no. 4, pp. 563–577, 2006. [Online]. Available: http://dx.doi.org/10.1109/TSMCC.2006.874022

[18] G. Morel, P. Valckenaers, J. Faure, C. E. Pereira, and C. Diedrich, "Manufacturing plant control challenges and issues," *Control Engineering Practice*, vol. 15, no. 11, pp. 1321–1331, Nov. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.conengprac.2007.05.005

[19] C. Sünder, A. Zoitl, and C. Dutzler, "Functional structure-based modelling of automation systems," *International Journal of Manufacturing Research*, vol. 1, no. 4, pp. 405 – 420, 2006. [Online]. Available: http://dx.doi.org/10.1504/IJMR.2006.012253

[20] O. Orozco and J. Martinez Lastra, "A Real-Time interface for Agent-Based control," in *Proceedings of the International Symposium on Industrial Embedded Systems (SIES '07)*, 2007, pp. 49–54. [Online]. Available: http://dx.doi.org/10.1109/SIES.2007.4297316

[21] M. Merdan, W. Lepuschitz, I. Hegny, and G. Koppensteiner, "Application of a communication interface between agents and the low level control," in *Proceedings of the 4th International Conference on Autonomous Robots and Agents (ICARA 2009)*, Feb. 2009, pp. 628–633. [Online]. Available: http://dx.doi.org/10.1109/ICARA.2000.4804006

[22] E. Gatt, "On three-layer architectures," in *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. AAAI Press, Mar. 1998, pp. 195–210.