Service-Oriented Agents for Collaborative Industrial Automation and Production Systems

J. Marco Mendes[1], Paulo Leitão[2], Francisco Restivo[1], and Armando W. Colombo[3]

[1] Faculty of Engineering – University of Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
{marco.mendes,fjr}@fe.up.pt
[2] Polytechnic Institute of Bragança, Quinta S^{ta} Apolónia, Apartado 134, 5301-857 Bragança, Portugal
pleitao@ipb.pt
[3] Schneider Electric Automation GmbH, Steinheimer Str. 117, D-63500 Seligenstadt, Germany
armando.colombo@de.schneider-electric.com

Abstract. Service-oriented Multi-Agent Systems (SoMAS) is an approach to combine the fundamental characteristics of service-oriented and multi-agent methods into a new platform for industrial automation. Several research works already targeted the connection of these technologies, presenting different perspectives in how and why to join them. This research focuses on available efforts and solutions in the area of SoMAS and explains the idea behind the service-oriented agents in industrial automation. A SoMAS system is mainly composed by shared resources in form of services and their providing/requesting agents. The paper also discusses the required engineering aspects of these systems, from the internal anatomy to the interaction patterns. Parameters of flexibility, reconfiguration, autonomy and reduced development efforts were considered and they should be the trademark of SoMAS. Aiming to illustrate the proposed approach, an example of service-oriented automation agents is given.

Keywords: Multi-agent Systems, Service-oriented Architectures, Industrial Automation.

1 Introduction

Current industrial automation and production systems are complex, heterogeneous in nature and require enormous development and maintenance efforts. The growing needs for more flexible and reconfigurable production systems has led to the development of new paradigms. One promising approach is to have a conglomerate of autonomous, reusable and co-operating entities (objects) that, under a functional point of view, are capable of dynamically interacting with each other to achieve both local and global objectives. When they are considered within a cross-layer infrastructure, like a manufacturing enterprise [1], some of them are embedded into automation devices and others are available in computing devices for more complex tasks. Attending to the nature and requirements of such systems, several software methodologies are being considered: the open standard of IEC 61499 to build functional block-based automation control systems, the more experimental Multi-Agent Systems (MAS) and more recently Service-oriented Architectures (SoA). Although one of the common objectives of these methodologies is to provide a sort of interoperability and flexibility over distributed automation systems, their background and specification is quite dissimilar.

From the research and development background of the authors on agent-based, service-oriented and component-based systems for automation (see for example the publication [2]), it was observed that the existing service-oriented entities that were specified, assimilate agents in terms of autonomy over a particular problem/resource and their “social” behavior. This observation motivated a research in the field of integrating multi-agent and service-oriented systems. This suggestion is not new since services are already part of agents’ specification, such as in FIPA (Foundation for Intelligent Physical Agents) [3], and agents are also present in standard documents of SoA methodologies [4]. Thereof, the under-considered elements (services in MAS and agents in SoA) are vaguely defined and have a more passive and customized role. Both agents and services have a very active function in the developed work for industrial automation, intersecting them to form the so-called Service-oriented Multi-Agent Systems (SoMAS). Most research dealing with the convergence are more focused on interoperability problems, semantic descriptions and simplification of problems with compositions, but the real nature, anatomy, behavior, interactions and autonomy of service-oriented agents is still less considered.

The concepts and studies introduced in this paper represents the authors’ experience in converging multi-agent and service-oriented systems in industrial automation and production systems, resulting in the form of service-oriented multi-agent systems. Following the introduction, section 2 reviews the application of multi-agent and service-oriented systems in industrial automation and discusses the integration of both paradigms. Section 3 introduces the adopted approach for service-oriented multi-agent systems and illustrates it with a simple example. Finally, section 4 resumes the conclusions and defines future work.

2 Background Information and Review of Literature

Service-oriented Architectures is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. This definition is according to the Reference Model for Service Oriented Architecture [4], which indicates an abstract framework for understanding significant entities and relationships between them within a service-oriented environment, and for the development of consistent standards or specifications supporting that environment. SoA is a means of organizing solutions that promotes reuse, growth and interoperability. Visibility, interaction, and effect are key concepts for describing the SoA paradigm. Note that although SoA is usually implemented using Web Services (WS), services can be made visible, support interaction, and generate effects through other implementation strategies. In a similar way, agent systems are frequently developed to be compliant to the IEEE FIPA standards. FIPA Abstract Architecture [3] includes a specification that defines architectural elements and their relationships.

2.1 Overview of Agent and Service Systems in Industrial Automation

Agent systems were introduced in automation and manufacturing when there was the vision of having distributable software components to resolve local tasks, comparable to the heterogeneous nature of the physical domain. Existing models of production, such as Flexible Manufacturing Systems (FMS) [5], Reconfigurable Manufacturing Systems (RMS) [6] and Collaborative Manufacturing Management (CMM) [7], are a basis for the application of multi-agent systems to solve and optimize the problems of production, due to the increase of introduced complexity [8]. Among the several publications reporting the application of MAS in automation and production systems, the reader can consult the reference [9] to retrieve more information on this topic.

Service-oriented systems are well known in the field of business and electronic commerce but in industrial automation systems (especially concerning distributed devices) it is a relatively new research area with promising results. The root of SoA fits well with collaborative automation, in the sense of autonomous, reusable and loosely-coupled distributed components. In automation domain, the vision of using SoA is to support the life-cycle needs in the context of agile and flexible manufacturing, addressing distributed, modular and reconfigurable automation systems which behavior is regulated by the coordination of services. The SIRENA Project [10] has contributed to strength the SoA-based automation by providing Web services at device level through the extension of the SoA paradigm into the realm of low-level embedded devices, such as sensors and actuators. Since then, significant research is going on, covering the engineering of such systems, including the process modeling, semantic description and collaboration. Research projects such as SOCRADES [11] and SODA [12] are contributing to develop extensions in service-oriented automation systems.

The integration of SoA with MAS in industrial automation and concretely its benefits are still unknown, but had started to be discussed. Being both currently the most promising concepts in this matter, the suggestion is that there are substantial benefits in converging both paradigms and technologies [13]. Other conclusions can be obtained from the application in other fields, as described in the next subsection.

2.2 Integration of Agents and Services

MAS and SoA are two paradigms that are based on separated models, but which also share common background requirements, features and applications.

The concept of service is frequently presented in publications about agent systems. In [14], services are used to provide a generic scheme to describe interactions between two agents. The authors of [15] propose an extension of the Web services-agent integration toolkit WS2JADE for Web services management with FIPA compliant multi-agent systems. The AgentWeb Gateway is an initiative for dynamic and seamless interoperation of multi-agent systems and Web services [16]. In [17], an agent-based service-oriented system architecture for manufacturing enterprise collaboration is presented. The authors of [18] explore how much the most common agent-oriented methodologies are also oriented to the development of services, comparing different agent-oriented methodologies under the point of view of the development of services. Integration of services is also discussed in [19], but rather presents available solutions in a more technological point of view. Agents have being widely studied in the way of creating compositions and orchestrations of services that can be understood as a simplification of services in a single one [20]. The power of reasoning provided by agents and the facility of semantic descriptions that are available in services is identified as a strong linkage to prove the usefulness of combining both solutions. As an association, the composition of (Web) services is similar to the planning problem that has been investigated extensively in artificial intelligence [21], including multi-agent systems.

In the same way that the word *service* is known in the "agent world", the terminology of *agent* is not new to SoA (see [4]). The main issue is that both paradigms have a distinguish evolution and traditional application fields; thereof they are more focused on their specific questions. According to [22], agent-based services are different from (Web) services, because they can support forming big services automatically and dynamically by the way of liberal coalition themselves in that each participant is autonomous.

The convergence of MAS and SoA can be discussed in different ways. From one side, the idea is to create interoperability mechanisms between the "agent world" and the "service world", using specific gateways/proxies that normally translate the semantic from one side to the other by the means of protocol translation (see [15] and [16]). Another possibility is to encapsulate single agents as services (agent-based service) and thus having a direct access to other services (see [17] and [23]). A completely different approach is the concept of service-oriented agents, discussed in this paper, that not only share services as their major form of communication, but also complement their own goals with external provided services (see [24]). Figure 1 depictures graphically the three approaches.

Some researchers discuss subjects such as intelligent services [25] or agent as services [23], but from our side a pertinent question is: should the service be intelligent or its provider/requester entity? A clear separation must be handled between the shared resource/action (service) and the entity that is responsible of the share. A simple example is a nurse offering a healthcare service to a patient. Who is intelligent? The healthcare service, the nurse or the patient? The answer would be the nurse and the patient, since the healthcare service is just a resource that is offered. This also demonstrates a strong linkage between services and the entities that provide/request them. Intelligent providers can decide over different steps of the service and requesters may able to negotiate previously before accepting the service, as an example. Agents may be the service's providers and requesters, since their definition fits well with this concept.

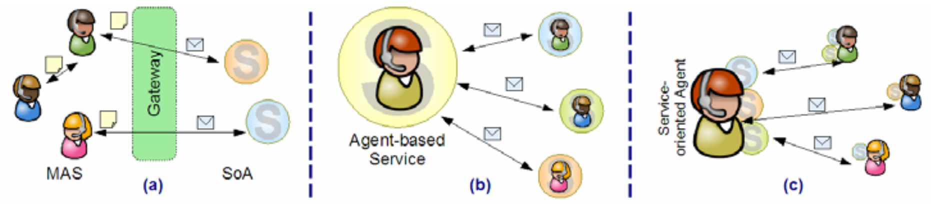

Fig. 1. Three commonly used approaches for integrating SoA and MAS: (a) interoperability gateway between MAS and SoA, (b) SoA encapsulating agents, (c) SoMAS

The challenge is thus the enrichment of SoA with some of the MAS characteristics (or vice-versa), to provide a more dynamic and flexible approach on how services are exposed and consumed, and how collaboration can be done using service-orientation. Cooperation is necessary and essential for multi-agent systems. In most of applications, especially in Internet-based distributed systems, service's request and offer are the main cooperation manners among agents, where agent either requests to get services or offers services for other agents. Such a system can be taken as service-oriented system [26]. The agent's characteristics of pro-action and autonomy, and agents' ability to negotiate commitments and deal with exceptions, are needed for the anticipated applications of service-oriented computing [23]. Therefore, agents can be used to build high-level models with flexible interaction patterns, while Web services are more suitable for solving interoperability problems among heterogeneous applications across enterprises [17].

However, issues regarding how they may be competing or complementary technologies remain open. Because of that, research involving agents and Web services is mainly focused on building improved semantics, communication languages and interaction protocols [27]. This represents one research direction where interoperability and the meaning of things are key issues to drive the use of fixed protocols into new directions (see [28]). The other one is reducing complexity: the concept of agents used in the Web service world is related to solve complex problems on the Web or as intermediaries' entities that make a complex organization externally accessible, and related to the way of simplifying the interface [20]. Reduced research has been seen in terms of the nature of service-oriented agents by the means of the necessity of requesting services, the conditions to provide services and their autonomy over a specific part of the overall (automation) problem.

3 An Approach for Service-Oriented Multi-agent Systems

The main requirement of modern industrial automation and production systems is to provide enough flexibility for automatic reconfiguration, able to respond promptly to changes in the environment and to client demands. To support the service-orientation of the systems that the authors were exploring, software components were specified and developed (see [2]). These service providing/requesting software components generally are used to represent mechatronic devices and enabled with a certain degree of autonomy to control the device, designated as Smart Mechatronic Components (SMeC). The interaction (or collaboration to reach global production objectives) is done via the access to the provided services. This work is reported in [2], where the reader can obtain more information.

The autonomy, mediation/representation and ability for collaborative processes that the previously explained components manifest are similar to what normal agents should have. An agent, according to [29] is a computer system situated in some environment, and that is capable of flexible autonomous actions in its environment in order to meet its design objectives. The flexibility of an agent is characterized by its responsiveness, pro-activeness and social behavior. The intelligence is still a matter of subjective appreciation, but according to the previous definition, the cited SMeCs exhibit the main features associated to an (intelligent) agent.

From this observation, a motivation rose in exploring the area of SoA and MAS paradigms to extract useful features and a new form of building automation systems. The adopted solution comes in form of a Service-oriented Multi-Agent System (SoMAS) that combines elements from both models, but also maintains a certain degree of retro-compatibility to its originators. The subsequent sections discuss the adopted approach based on the research work and the existing requirements to build service-oriented agents for industrial automation.

3.1 The SoMAS Approach

The proposed approach is characterized by using a set of distributed autonomous and cooperative entities, the agents, using the SoA principles. The development of such SoMAS, presenting differences in relation to the traditional MAS, requires an engineering development framework that supports the new functionalities. More specifically, it describes a model for the development of a MAS oriented by the offer and request of services, in order to fulfill industrial and production systems goals. The approach is different from traditional MAS in the form of:

– *Agents are service-oriented*: individual goals of agents may be complemented by services provided by other agents. Internal conditions of agents can be offered as services for others to be requested.
– *Features inherited from service-orientation*: a main feature is the re-usability of services of an agent that can be used and applied not only for a single type of a problem for what it was developed, but also be re-used for different tasks.

The model provides the general guidelines to specify SoMAS for resolving problems and achieve goals in industrial automation and productions. Figure 2 shows the conceptualization based on the different key concepts and an overview of the system's concept. A brief description about each key concept follows:

– *Domain - Industrial Automation and Production*: the domain of the model is applied for industrial automation and production, made of software and hardware components.

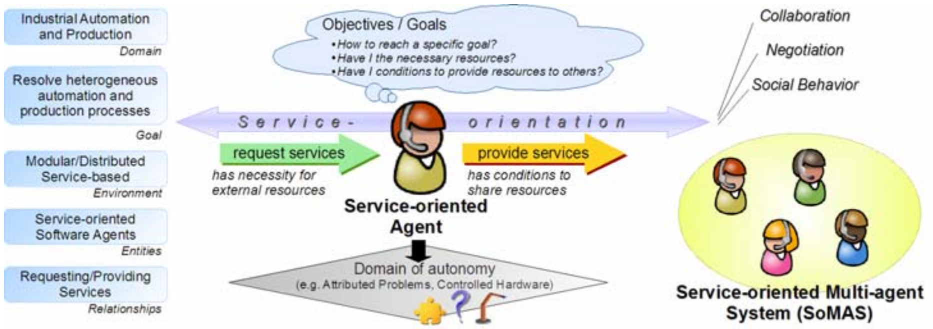
Industrial Automation and Production
Domain
Resolve heterogeneous automation and production processes
Goal
Modular/Distributed Service-based
Environment
Service-oriented Software Agents
Entities
Requesting/Providing Services
Relationships
Objectives / Goals
•How to reach a specific goal?
•Have I the necessary resources?
•Have I conditions to provide resources to others?
Service- orientation
request services
has necessity for external resources
provide services
has conditions to share resources
Service-oriented Agent
Domain of autonomy
(e.g. Attributed Problems, Controlled Hardware)
Collaboration
Negotiation
Social Behavior
Service-oriented Multi-agent System (SoMAS)

Fig. 2. Model and conceptualization of SoMAS

– *Goal - Resolve heterogeneous automation and production processes*: since modern and flexible systems are made of and involve heterogeneous and concurrent activities, the model is targeting the management and resolution of processes for the automation of devices and also the production of goods. Processes may comprehend simple statically descriptions of work-plans to fulfill a given goal or dynamically constructed collaborations for a more flexible approach.
– *Environment - Modular/Distributed, Service-based*: in the system there are software and hardware components. Software components are generally agents that have specified tasks and also may represent/control hardware components. Resources and functions of agents and other software components are exposed as services.
– *Entities - Service-oriented Software Agents*: a software agent is an entity that may act for a user or device and has the ability to achieve their own goals. Several additional concepts include the degree of autonomy of agents and also their learning and intelligence capabilities. A multi-agent system is used when there are complex objectives made of separable individual problems that can be resolved by single agents and their collaboration, reducing the complexity of the overall problem.
– *Relationships - Requesting/Providing Services*: problems and sub-problems are solved by agents that should collaborate. The collaboration is achieved by providing services by agents (when they have the conditions) and their requesting (when agents have needs).

Since this specification represents only a general model, the communication and the development technology is open. This means that any suitable technology can be used to develop the concepts presented by the model. For example, Web services technologies can be adapted to the resource sharing via services and C++ or Java languages can be used to develop the agents.

3.2 Engineering Aspects

Agents will not "work" by their own unless some basic specifications for the engineering related to the domain are considered. Several additional topics are related to the engineering of systems that are specified and developed according to the proposed concept. From the other side and since the model is technology independent, the development of consequent service-oriented multi-agent systems can be done using the current available agent development frameworks and service-oriented systems.

The FIPA Abstract Architecture [3] explicitly avoids the issues related to the internal structure of an agent. It also largely defers details of agent services to more concrete architecture documents. The same openness is also exhibited by the proposed approach, thus each agent may be implemented internally independently and differently (i.e. using different technologies, e.g. different decision techniques and different programming languages). The only requirement is that it should share its functions as services and obey to the protocols of communication and processes that were established. This flexibility may contribute for the development of customized agents for diversified tasks. A suggestion for the internal specification of these service-oriented agents is the adoption of a modular approach as the one described in [30]. It introduces an anatomical-like structure for the development of functional and reusable modules of a component/agent, part of a service-oriented automation system.

One of the most important issues is the interaction among entities. A specification for interaction patterns for service-oriented entities was elaborated and documented in [31], similar to the interaction approach of agent systems. The proposed solution defines a set of interaction phases for the use of services: Discovery, Negotiation, Operational and Termination. Each one of these phases is managed by specific ports of the service. For example, if a requester would like to access a transportation service provided by some agent, it would first enter in the discovery phase and then proceed to the negotiation phase, accessing a specific negotiation port of the discovered transportation port. After successful negotiation, the operation of the services is activated by the operational port. Termination routines can then be accessed via the termination port.

Besides the interaction, the following engineering aspects have also to be considered in the engineering framework:

– *Description of knowledge and semantics*, i.e. agents may understand what is the information associated to services and other elements.
– *Centralization/decentralization of entities (agents) and tasks*: due to the flexibility of the model, implemented systems may adopt a more centralized approach (e.g. master/slave agents) or a completely decentralized distribution (in a more collaborative fashion).
– *Description of processes* that describe the behavior of agents and usage of services can be used as a complement for coordination of activities.
– Services may be *aggregated/associated*, simplifying the external access to resources.
– A way to *address and discover* services and agents.
– *Autonomy and pro-activity* of agents should be considered in the sense of not only providing services, but also in the ability of requesting them and having “autonomy” of decision and control over some part of the problem.
– *Design and agent development tools,* i.e. providing an environment and framework for easy development and monitoring of agents and services.
– *Security aspects.*

Depending on the type of implementation, other requirements for engineering may also be necessary, besides the ones previously referred.

3.3 Example of a Service-Oriented Agent

There are several examples and applications that can be given based on previous explained concept. A traditional manufacturing scenario includes several components that can be distinguished: processing machines, transport systems, warehouses and the products itself. Considering a multi-agent system applied to this scenario, software agents would represent the several elements in the system and act over them.

A concrete example agent is applied as a mediator of a virtual unidirectional conveyor to transport pallets from one port to another (see Figure 3). As such, the agent has several functional modules: Communication and Device Interface. These modules are part of the agent’s anatomy and permit the agent to communicate via service-orientation and access the physical conveyor, respectively. In its core the agent provides a service (*Transfer* service) because it has the conditions to do so. But to proceed to a specific transfer of pallets, that is its goal, it also needs to transfer the

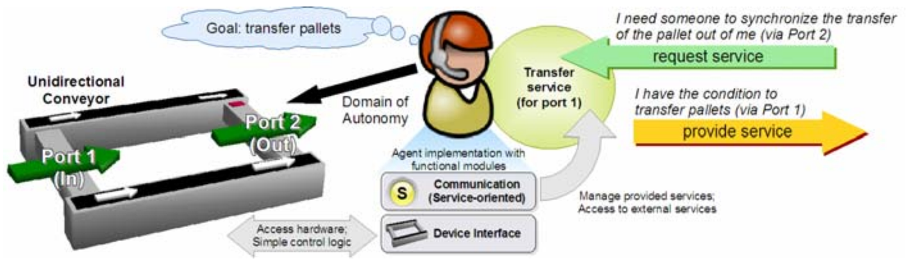
Goal: transfer pallets
Unidirectional Conveyor
Port 1 (In)
Port 2 (Out)
Domain of Autonomy
Transfer service (for port 1)
I need someone to synchronize the transfer of the pallet out of me (via Port 2)
request service
I have the condition to transfer pallets (via Port 1)
provide service
Agent implementation with functional modules
Communication (Service-oriented)
Device Interface
Manage provided services; Access to external services
Access hardware; Simple control logic

Fig. 3. Example of a service-oriented agent (mediator of a bidirectional conveyor)

received pallet to a destination. Consequently, it will request a concrete service from a neighbor device that provides the necessary capabilities. This sort of interactions by complementing own objectives with service-orientation, results in a collaboration in benefit of the overall production and building complex nets of automation systems.

The several keywords used previously and that represent the features of SoMAS are explained below using the example of Fig. 3:

- *Modularity and autonomy*: automation and production problems are subdivided and one agent is autonomous in charge of part of it. In the example, the agent is responsible for the management and objectives of the unidirectional conveyor.
- *Interoperability and orientation*: agents are service-oriented and this characteristic strengths the interoperability between different entities in the system. In the example, the agent is able to create its own services, but in order to respect the global goals it must be able to use other services that are in the system. For the full function of the conveyor, services provided by neighbor transport systems should be used by the local agent to transport pallets to the required designations.
- *Transparency and reduced development efforts*: agents can be developed using different methodologies and technologies, but should obey to interaction patterns that are defined and standard to each other. Proposed methodologies such as the internal anatomy for agents [30] and interaction patterns [31] were specified to reduce development efforts and increase functionality.
- *Flexibility and reconfiguration*: system flexibility is introduced by the autonomy and modularity of agents, which can be used for reconfiguration purposes (e.g. local adjustments without stopping the whole system, environmental changes, and modification of production objectives).
- *Integration*: the use of service-oriented principles over multi-agent systems simplifies the enterprise vertical integration, namely into business levels (commonly operating in a service-oriented way).

3.4 Application of Agent and Service Frameworks

The development of a framework based on the present concepts considered several existing implementations. JADE (Java Agent DEvelopment Framework), a widely used framework in the academic area for R&D purposes, is a software framework written in the Java language for the development of FIPA-compliant agents and agent-based systems. Other examples of frameworks for developing agent systems can be referred, such as FIPA-OS, JACK and Mobile-C. One of the main problems of the current agent frameworks is that they have always some sort of centralized management (e.g. containers, agent management system, and directory facilitator) and are not truly distributed and decentralized.

For SoA, it is more difficult to find a complete framework to develop service-oriented systems, being usually only available a set of tools for the development of specific features of the architecture. For example, the SoA for Devices (SOA4D) open toolkit can be used for the development of service-oriented software components in Device Profile for Web Services (DPWS). Therefore, it is only able to create the necessary structures for providing and requesting services, including also the discovery, addressing, etc. One point in favor, besides creating a basis for service-oriented communication, is the possibility of creating real (100%) distributed systems without central management (such as it happens in the current agent frameworks). For example the dynamic discovery permits localization of services without having a central yellow pages server. Other characteristic is the possibility of dynamically configure devices and services using dynamic deployment.

Since the authors were already working on a framework for service-oriented components [32], the previous development approach was also considered for developing service-oriented agents, and to cover some of the previous discussed limitations present in SoA and agent frameworks. Some required features include development on C/C++ languages (targeting multi-platform and specially embedded devices), source code availability and extendable, modular and functional body of agents, and a certain degree of autonomy (not only from the concept of autonomous agent, but also from the point of view of the final software component). Considering the requirements, the existing *Continuum* project was used [32]. The *Continuum* project aims the development and support of the engineering processes in industrial automation systems, considering also the requirements described in this document.

4 Conclusions and Future Work

This paper introduces an approach for the development of distributed intelligent systems that exhibit modularity, flexibility, re-configurability and interoperability features, combining the best features of MAS and SoA paradigms. For this purpose, research was done in the direction of SoMAS, being explained the idea of service-oriented agents. The SoMAS approach introduces main advantages over the traditional MAS and SoA paradigms by combining the best features of MAS and SoA paradigms enhancing the achievement of autonomy, modularity, flexibility and interoperability, and consequent reconfiguration that are required by distributed control systems. Future work is related to enhance the specification of SoMAS and to continue with the development of the framework to be able to specify service-oriented agents for real and virtual automation scenarios. An important question to be resolved is to adopt the compliance with the FIPA specifications.

Acknowledgments

The authors would like to thank the European Commission and the partners of the EU IST FP6 project "Service-Oriented Cross-layer infrastructure for Distributed smart

Embedded devices” (SOCRADES), the EU FP6 "Network of Excellence for Innovative Production Machines and Systems” (I*PROMS), and the EC ICT FP7 project “Cooperating Objects Network of Excellence” (CONET) for their support.

References

1. Deen, S.M.: Agent Based Manufacturing: Advances in the Holonic Approach. Springer, Berlin (2003)
2. Mendes, J.M., Leitão, P., Colombo, A.W., Restivo, F.: Service-oriented control architecture for reconfigurable production systems. In: Proceedings of the 6th IEEE International Conference on Industrial Informatics, pp. 744–749 (2008)
3. FIPA Abstract Architecture Specification. Standard of the Foundation for Intelligent Physical Agents (2002), `http://www.fipa.org/specs/fipa00001`
4. Reference Model for Service Oriented Architecture 1.0. OASIS Standard (October 12, 2006), `http://docs.oasis-open.org/soa-rm/v1.0`
5. Tombak, M., De Meyer, A.: Flexibility and FMS: an empirical analysis. IEEE Transactions on Engineering Management 35(2), 101–107 (1988)
6. Mehrabi, M.G., Ulsoy, A.G., Koren, Y.: Reconfigurable manufacturing systems: Key to future manufacturing. Journal of Intelligent Manufacturing 11(4), 403–419 (2000)
7. Gorbach, G., Nick, R.: Collaborative Manufacturing Management Strategies. White paper, ARC Advisory Group (2002)
8. Tnazefti-Kerkeni, I., Arantes, L., Moalla, M.: An agent-oriented architecture for F.M.S. control/monitoring. In: Proceedings of 2003 IEEE Conference on Control Applications, vol. 2, pp. 1024–1028 (2003)
9. Marik, V., McFarlane, D.: Industrial adoption of agent-based technologies. IEEE Intelligent Systems 20(4), 27–35 (2005)
10. Jammes, F., Smit, H.: Service-oriented architectures for devices - the SIRENA view. In: Proceedings of the 3rd IEEE International Conference on Industrial Informatics, pp. 140–147 (2005)
11. Taisch, M.: The Socrades European project (Service-Orientated Cross-layer InfRAstructure for Distributed Smart Embedded Devices). In: Second World Congress on Engineering Asset Management (EAM) and The Fourth International Conference on Condition Monitoring (2007) (presentation)
12. Mensch, A., Depeisses, F.: SODA Technical Framework Definition (2007), `http://www.soda-itea.org/Documents/Specifications/1176731057.06.html`
13. Ribeiro, L., Barata, J., Mendes, P.: MAS and SOA: Complementary Automation Paradigms. In: IFIP International Federation for Information Processing, vol. 266, pp. 259–268. Springer, Boston (2008)
14. Sesseler, R.: Building agents for service provisioning out of components. In: Proceedings of the fifth international conference on Autonomous agents, pp. 218–219. ACM Press, New York (2001)
15. Nguyen, X.T., Kowalczyk, R.: Enabling agent-based management of Web services with WS2JADE. In: Proceedings of the Fifth International Conference on Quality Software, pp. 407–412 (2005)
16. Shafiq, M.O., Ali, A., Farooq Ahmad, H., Suguri, H.: AgentWeb Gateway - a middleware for dynamic integration of multi agent system and Web services framework. In: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, pp. 267–268 (2005)

17. Shen, W., Ghenniwa, H., Li, Y.: Agent-Based Service-Oriented Computing and Applications. In: Proceedings of the 1st International Symposium on Pervasive Computing and Applications, pp. 8–9 (2006)
18. Cabri, G., Leonardi, L., Puviani, M.: Service-Oriented Agent Methodologies. In: Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 24–29 (2007)
19. Greenwood, D., Lyell, M., Mallya, A., Suguri, H.: The IEEE FIPA approach to integrating software agents and web services. In: Proceedings of the 6th international joint conference on autonomous agents and multi-agent systems. ACM, New York (2007)
20. Walton, W.: Agency and the Semantic Web. Oxford University Press, USA (2006)
21. Singh, M., Huhns, M.: Service-Oriented Computing: Semantics, Processes, Agents. John Wiley & Sons Ltd., England (2005)
22. Yang, X., Feng, Z., Xu, G.: The Automatic Formation of Agent Societies: A Service-Driven Approach. In: Proceedings of the International Conference on Advanced Language Processing and Web Information Technology, pp. 556–561 (2008)
23. Huhns, M.N.: Agents as Web services. IEEE Internet Computing 6(4), 93–95 (2002)
24. Paulino, H., Lopes, L.: A service-oriented language for programming mobile agents. In: Proceedings of the fifth International joint Conference on Autonomous Agents and Multi-agent Systems, pp. 1294–1296. ACM Press, New York (2006)
25. Shen, W., Li, Y., Hao, Q., Wang, S., Ghenniwa, H.: Implementing collaborative manufacturing with intelligent Web services. In: Proceedings of the Fifth International Conference on Computer and Information Technology, pp. 1063–1069 (2005)
26. Xinjun, M., Gang, W., Huaiming, W.: Cooperation models for service oriented multi-agent system. In: Proceedings of the 2004 ACM symposium on Applied Computing, pp. 510–511. ACM Press, New York (2004)
27. Ramírez, E., Brena, R.: Multi-Agent Systems Integration in Enterprise Environments Using Web Services. In: Agent and Web Service Technologies in Virtual Enterprises, Information Science Reference, pp. 174–189. Hershey, New York (2008)
28. Huhns, M.N., Singh, M.P., Burstein, M., Decker, K., Durfee, E., Finin, T., Gasser, T.L., Goradia, H., Jennings, P.N., Lakkaraju, K., Nakashima, H., Parunak, H., Rosenschein, J.S., Ruvinsky, A., Sukthankar, G., Swarup, S., Sycara, K., Tambe, M., Wagner, T., Zavafa, L.: Research directions for service-oriented multiagent systems. IEEE Internet Computing 9(6), 65–70 (2005)
29. Jennings, N.R., Wooldridge, M.: Applications of intelligent agents. In: Agent technology: foundations, applications, and markets, pp. 3–28. Springer, New York (1998)
30. Mendes, J.M., Sousa, J., Leitão, P., Colombo, A.W., Restivo, F.: Event Router-Scheduler for the Modular Anatomy of Service-oriented Automation Components. In: Proceedings of the 6th CIRP International Conference on Intelligent Computation in Manufacturing Engineering (2008)
31. Mendes, J.M., Rodrigues, A., Leitão, P., Colombo, A.W., Restivo, F.: Distributed Control Patterns using Device Profile for Web Services. In: Proceedings of the 12th International IEEE EDOC Conference Workshop (2008)
32. Mendes, J.M., Bepperling, A., Pinto, J., Leitão, P., Restivo, F., Colombo, A.W.: Software Methodologies for the Engineering of Service-Oriented Industrial Automation: The Continuum Project. In: Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference (to appear, 2009)