2013 IEEE International Conference on Systems, Man, and Cybernetics

A Conceptual Architecture Based on Intelligent Services for Manufacturing Support Systems

César A. Marín[*], Lars Mönch[§], Paulo Leitão[‡], Pavel Vrba[†], Daria Kazanskaia[¶], Vadim Chepegin[||], Liwei Liu[*], Nikolay Mehandjiev[*]
[*]Service Systems Group, MBS, The University of Manchester, Manchester M15 6PB, United Kingdom
{*forename.surname*}@manchester.ac.uk
[†]Department of Cybernetics, Czech Technical University, Karlovo namesti 13, 121 35 Prague 2, Czech Republic
[‡]Polytechnic Institute of Bragança, Campus Sta Apolónia, Apartado 1134, 5301-857 Bragança, Portugal
[§]Enterprise-wide Software Systems, University of Hagen, 58097 Hagen, Germany
[¶]Smart Solutions Ltd, Moskovskoye shosse 17, 443013 Samara, Russia
[||]TIE Nederland B.V, De Corridor 5d, Breukelen 3621 ZA, The Netherlands

***Abstract*—The contribution of this paper is a conceptual architecture based on intelligent services for manufacturing support systems. Current approaches cannot cope with the dynamism inherent in the manufacturing domain, where unexpected and drastic changes occur at any time and affect cross-boundary systems. Our approach alleviates these issues and focuses on the need for supporting real-time automatic negotiation, planning, scheduling and optimisation, intra and across factories. We present an execution example of our approach in the aerospace industry, instantiating the conceptual architecture using different technologies. With this we push forward a new definition of Intelligent Services and indicate future work in the area.**

***Index Terms*—enterprise service bus, holonic agent systems, manufacturing support systems**

I. Introduction

Globalisation and speed of technological change have changed traditional manufacturing to a highly dynamic and complex domain [1], [2], which increasingly relies to information technology to coordinate manufacturing activities and provide information to the right place in the process. The underlying software architectures and technologies have so far been fairly difficult to change and thus have provided inadequate support for the dynamism and complexity of the domain. In the last 10 years service-based systems have gained recognition as a technology which can address this issue using loose coupling, platform independence and standardisation of interfaces [3]. Yet the level of dynamism supported by services is limited by the incremental approach to their application, which inhibits the effective use of features such as dynamic composition and distributed coordination [3].

Within high-volume manufacturing industries such as automotive manufacturing, the dynamism is encountered at initial stages of design and prototyping, whilst the stage of mass production is characterised with relative stability. Small lot production such as airplanes and ships, on the other side, is characterised with high degrees of customisation of the individual products. The supply chains are highly distributed, with a large number of suppliers spread across countries and even continents. This creates uncertainty of deliveries during the stages of ramp-up and production. The high degree of customisation and dynamism in supply chains means that the need of flexible software to provide information, coordinate activities and support decision-making is especially acute within the domain, making traditional approaches such as ERP and MES no longer adequate.

Attempts to use software agents as a basis for satisfying these needs, often in combination with Web service technologies exist within the domain, cf. [4]. They use the inherent capabilities of such technologies to deal with dynamism, yet compromise computational requirements and interoperability. We believe that a new, conceptual fusion of the two approaches is the way forward to delivering the degree of flexibility required by the domain, and creating an integrated approach of Intelligent Services.

In this paper we present a conceptual architecture based on a new definition of Intelligent Services for the support of information systems in manufacturing (Sect. IV), in particular meeting the requirements of small lot production (Sect. III). We present a running example (Sect. V) with three different instantiations of the conceptual architecture using different technologies. With this example we emphasise the novelty that using this new definition of Intelligent Services we can deal with complexity and dynamism in this domain without committing to a unique technology. Finally, we compare our approach with others (Sect. VI) and indicate future directions in this domain (Sect. VII).

II. Background: Agents, Services and their Combinations in Manufacturing

Requirements in manufacturing impose the need to have more flexible, adaptive and robust information systems, based on the distribution and decentralisation of control and decision-support. The multi-agent system (MAS) paradigm [5] meets these requirements by introducing a concept for modelling complex systems as community of autonomous entities — agents— that execute their activities in parallel and in the decentralised manner. Each agent, owning only a partial knowledge and limited set of skills, is able to communicate with other agents in order to jointly reach the global-level objectives. In the manufacturing domain, an agent can represent either a physical resource, such as a robot, conveyor belt, product, worker, etc. or a logical object, for example a scheduler, order or invoice. Due to the independence of a central control point, the industrial MASs are proved to have the capability to respond quickly to disturbances and adapt to changing conditions. The distributed decision making, along with the interactions between individuals, lead to the emergence of the "intelligent" global behaviour that is more than a mere sum of the behaviours of individuals [5].

Several MAS solutions have been developed and deployed into real manufacturing systems. For example, the Production 2000+ system installed at Daimler Chrysler's factory plant producing engine cylinder [6]; the agent-based system at SKODA Auto, in Mladá Boleslav, Czech Republic, for job scheduling at the engine assembling workshop [7]; a more detailed survey along these lines can be found in [4].

Syntactic and semantic interoperability in distributed heterogeneous environments is crucial. Yet MAS itself does not offer sufficient means for coping with this issue. The interoperability within a single agent system and even among multiple agent systems can be achieved by adopting suitable standards such as those provided by the Foundation for Intelligent Physical Agents (FIPA) [8]. FIPA produced a set of standards aimed at different aspects of agent-oriented software engineering such as agent management, agent communication, and agent message transport. The high popularity of JADE agent platform [9], which consistently applies FIPA standard, contributed to the proliferation of this standard in the agent developers community. Notwithstanding, FIPA has not been actively developed further in the last ten years. Thus it does not address the challenges of MAS integration with modern software engineering approaches such as service-oriented architecture (SOA).

The SOA paradigm is a way of building distributed software systems based on the concept of request and provision of reusable and interoperable software components —services. A service is a software component that encapsulates the business/control logic, responding to a specific request. A main concern in service-oriented systems is how the services "play" together, emerging the concepts of orchestration and choreography. Orchestration is the practice of sequencing and synchronising the execution of services [10]. Choreography, on the other hand, considers the rules that define the messages and interaction sequences that must occur in order to execute a given process through a particular service interface.

Recently, (Web) services are injected with "intelligence" by fusing them with MAS. The concept of Intelligent (Web) Service encompasses Web services embedded with MAS features, in particular collaboration, adaptation, and communication [11]. By doing so, the functionalities of MAS systems are enhanced and some of the limitations are overcomed in terms of interoperability and IT-vertical integration. In spite of being based on the same concept of providing a distributed perspective to the system, MAS and SOA present some important differences, namely in terms of computational requirements and interoperability (see [12] for a deeper analysis). These differences highlight the complementary aspects of the two paradigms, suggesting the benefits of combining them.

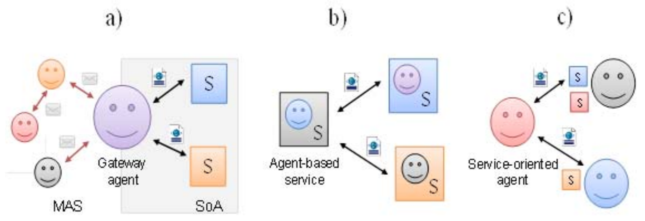

Figure 1. Integrating SOA and MAS (adapted from [14]).

This suggestion is not new since services are already part of the agents specification, e.g. in the FIPA specification [8], and agents are also present in standard documents of SOA methodologies, e.g. in the OASIS standard [13]. Therefore, the under-considered elements (services in MAS and agents in SOA) are vaguely defined and have a more passive and customised role. The combination of MAS and SOA has been performed in different ways, as shown in Fig. 1.

The first option (Fig. 1a) considers gateways to provide translation between the agent world and the service world. This concept is implemented for example by a Web Services Integration Gateway plug-in to the JADE platform. Through the Gateway agent the system offers transparent bi-directional transformations between FIPA compliant agents services and Web services employing the WSDL (Web Services Description Language)/SOAP (Simple Object Access Protocol)/UDDI (Universal Description Discovery and Integration) stack [9]. The second option (Fig. 1b) considers the encapsulation of single agents as services (agent-based services), enabling direct access to other services and permitting the addition of agents' skills, improving services capabilities. The last option (illustrated in Fig. 1c) is characterised by the use of a set of distributed autonomous and cooperative agents that use the SOA principles to fulfil the system goals.

The service-oriented MAS approach is different from the traditional MAS mainly because agents' individual goals may be complemented by services provided by other agents, and the internal functionalities of agents can be offered as services to others agents [14]. For example, [15] uses a model-driven approach that combines SOA and MAS to model a segment of a production chain in the steel industry. [16] proposes the integration of legacy systems by the encapsulation of its features by agents.

III. Requirements of low-volume manufacturing

Nowadays, customer needs have been increasingly diversified. Thus a large variety of sophisticated products has to be managed in current and future production sites to meet the needs. In particular, the large-scale product industry results in low-volume, high customisation manufacturing where an efficient utilisation of labour and machinery creates significant challenges [17], [18]. For example, aerospace companies often develop a large number of different airplane models to satisfy the requirements of customers. This type of production also leads to several challenges for engineering, increasing the effort for production planning and control. Frequently occurring errors demonstrate that product design, the production lines, the suppliers, IT, and logistics are not ready at the very first products of a series. Low-volume, high variety production in domains like aerospace or shipbuilding is often managed by companies that offer value-added services. They mainly deal with the labour-intensive assembly of parts and components that are manufactured by highly specialised suppliers. The supplier network has to be planned and controlled in an appropriate way.

Overall, the low-volume, high customisation production leads to the following requirements for a supporting information system:

R1. Human initiative at different organisational levels: In contrast to a fully-automated computation of production planning and control instructions, the production in the low-volume production is labour-intensive [19]. Consequently, some decision support has to be offered to the staff at the production planning and control department, but at the same time, the operational decisions made by assembly line managers, station managers, and workers have to be supported. This requires the ability to modify production plans and schedules manually. Therefore, appropriate human-machine interfaces are highly desirable.

R2. Distributed decision-making based on local data: Because of the physical decomposition of the overall production system including geographically distributed facilities with different assembly lines, each of them containing several assembly stations [20], there is a need for distributed decision-making based on the local data available. This requirement is also supported by the need to solve production planning problems including suppliers and various scheduling problems at different points of time.

R3. Negotiation-based coordination of schedules: From a production planning and control point of view, a single project is created for each customer order [19], [21]. The corresponding network plan is used to determine milestones for each customer order. These internal dates are used to monitor the progress of the order. The station manager decomposes the work packages for his station into individual work orders that are assigned to groups of workers with skills appropriate to carry out the tasks of the work orders. From a scheduling point of view, the scheduling problem on the task level can be modelled as a multi-mode resource constraint project scheduling problem (RCPSP) (cf. [19], [21]). A certain set of modes is assigned to each task. A mode is given by a specific number of workers with prescribed skills. The processing time to carry out the task depends on the number of workers. The overall number of workers has to be smaller or equal than the number of workers that are totally assigned to a station at any point of time. The multi-mode RCPSP requires that a certain number of workers is given for each station in a certain period of time. This number can be determined by techniques that are known as assembly line balancing (cf. [18], [22]). Note that the horizon for the multi-mode RCPSP can be rather short, i.e. a shift or several shifts, while the horizon for the second problem might be much longer, i.e. some weeks, months, or one year. Even a decomposition of the overall horizon in periods might be useful for the second problem. A single multi-mode RCPSP can be formulated and solved for each station. This leads to a a set of schedulers that have to be coordinated by the planner that is responsible for line balancing. Negotiation-based approaches are well known to be beneficial in solving RCPSP-type problems (cf. [23]).

R4. Connections to legacy systems: Appropriate integration with external systems such as Enterprise Resource Planning (ERP) systems or Manufacturing Execution Systems (MES) is crucial because these systems provide the data for decision-making during production planning and scheduling. But their functionality is limited with respect to the production planning and control tasks to be performed in the large-scale product industry [21], [22].

R5. Frequent updates to scheduling and (re)planning knowledge: Provision of production planning, scheduling and rescheduling functionalities should be able to cope with the frequent changes and other disturbances typical of complex assembly processes. Examples of these changes include changes of products, technologies, equipment, and toolsets at system runtime [20].

Traditional information systems for manufacturing and logistics like ERP and MES tend to be not sufficient in this dynamic, highly complex environment with a large number of disturbances, especially for near real-time decision making. Although the MAS and SOA paradigms support most of the domain requirements, ensuring interoperability between MAS and SOA tends to be restricted by their implementation.

IV. Conceptual Architecture as a Solution

In response to the requirements outlined in the previous section, we have developed a conceptual architecture for the support of small lot production manufacturing. We call it Intelligent Enterprise Service-based Bus (iESB). The architecture is based on **Intelligent Services** defined as *independent pieces of software that are expected to provide a particular result, either produced by the Intelligent Service itself or by requesting support from other Intelligent Services*. The major difference between an Intelligent Service and an Intelligent Web Service as in [11] is that the latter considers the fusion between Web services and software agents as the means to inject "intelligence" into Web services, focusing on the implementation details rather than on the concept. We argue that Intelligent Services are not bound by the underlying technology and can be instantiated by agents, Web services or any of their combinations.

Figure 2 depicts our architecture and internal components as Intelligent Services, namely an enterprise service bus, gateway, process manager, planner, scheduler, peer-2-peer layer,

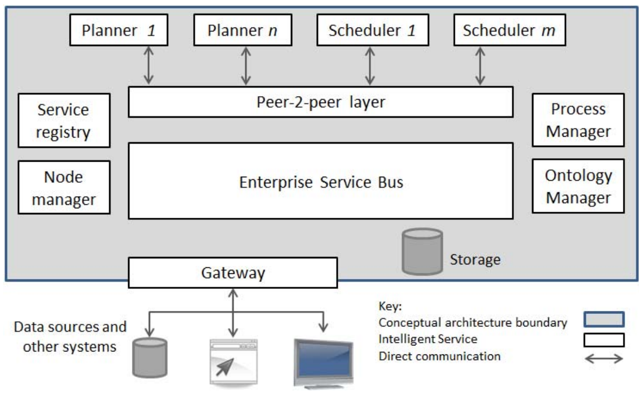

Figure 2. Intelligent Service based conceptual architecture for supporting low-volume production manufacturing.

ontology manager, service registry and node manager. We describe these Services below and indicate how they fulfil the requirements in Sect. III.

a) Supporting distributed intelligent processing: Requirements **R1** and **R2** from Sect. III suggest that an effective support system should facilitate coordination of activities across physical distances whilst reasoning with localised knowledge. This stipulates using intelligent distributed processing as implemented by software agent technology, integrating local reasoning with distributed coordination abilities. The enterprise context of application postulates the need for guaranteed performance and message delivery, message logging and event subscription, which are covered by enterprise service bus architectures. We therefore use *Intelligent Services* and integrate them with an *Enterprise Service Bus (ESB)* as a key Service to alleviate communication and interoperability issues [24] among Intelligent Service implementations. Messages are sent by Services to each other via the ESB. Services are also able to register an interest in message types so that they get notified when events of their interest within the system occur.

b) Supporting negotiation-based coordination of schedulers: In Sect. III, **R3** established that effective support systems should coordinate localised and global replanning and scheduling activities using negotiation and other agent-based coordination techniques. Within the architecture, we need to support this by allowing different schedulers and planners to coordinate their operation at a level conceptually higher than the ESB, although the specific messages may end up as being channelled through the ESB to use the guaranteed delivery mechanisms there. This is provided by the dedicated *Peer-2-peer Layer* which handles negotiation communications at a higher level of abstraction. Note than this Service only supports the Scheduler and Planner services, however this does not limit the latter from using other available services in the architecture. The *Planner and Scheduler* services optimise the execution of process instances according to available resources, timing, constraints, and unexpected events. The difference between the Planner and the Scheduler is the time horizon: the Scheduler deals with imminent tasks and next shift tasks, whereas the Planner deals with tasks weeks and months in advance. In both cases, unexpected events such as sudden increase of process instances, off-sick staff, broken machines, etc., are taken into account for rescheduling and adaptation. There may exist more than one of these Services depending on the factory configuration and the different types of processes to handle. Further details about the internal operation of these Services can be found in [25]. The final component relevant to this requirement is the *Process Manager*, which deals with the definition and management of manufacturing processes within the factory as well as definition and general allocation of resources such as staff, tools, machines, etc. Process instances are initiated by this Service according to how the user defines the processes themselves.

c) Connecting to legacy systems: To implement the requirement **R4** about the need to connect to a number of diverse external systems, the *Gateway* service functions as the interface for exchanging data with external data sources and systems such as ERP, MES, SCADA, etc. The gateway not only pulls information but also pushes data into external sources as needed. The availability of information from external sources and systems is notified to registered Services via the ESB for potential consumption.

d) Frequent update of planning and rescheduling knowledge: This requirement (**R5** from the previous section) captures the essence of the rapid change of technology, production and organisations which characterises global manufacturing markets. To cope with this, we need to design the architecture as a system which is open to new planning and scheduling techniques, new interface modules, new connections to legacy systems, and, last but not least, new knowledge about the domain and the production processes. The architecture thus comprises components such as the *Node Manager* and *Service Registry*. The Node Manager establishes initial connectivity between any two architecture instances. That is, Fig. 2 can be instantiated in different geographical locations, e.g. different factories. Then the Node Manager allows different running instances to communicate and share Services from one instance to the other. This procedure effectively converts each instance into a node of a system geographically distributed. The Service Registry keeps track of available Services at a node in the distributed system. It coordinates with the Node Manager for indicating availability of Services at a node on different, connected nodes. To handle the domain changes, the proposed architecture relies on ontology for modelling a variety of concepts. A first version of the ARUM ontology can be found in [26]. This way, the approach benefits from formal and machine-interpretable semantics, semantic interoperability and consistency, as well as inference of knowledge not explicitly contained in the models. The *Ontology Manager* Service uses a unified ontological approach for modelling domain knowledge, real-world entities, policies and operational workflows, providing to the proposed architecture with several advantages. The most fundamental is that said concepts are treated in an integrated manner. This way, the model can effectively capture their dependencies, while allowing for the enforcement

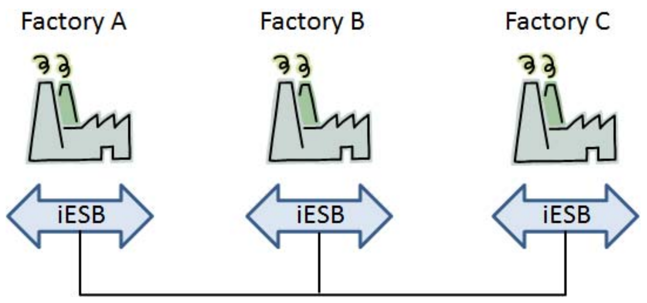

Figure 3. Distributed iESB across three factories.

of interdependencies at all phases of their lifecycle.

A discussion on the advantages and limitations of the conceptual architecture are included in Sect VI. The following section provides a running example of the architecture highlighting the functionality of the aforementioned Services.

V. Running Example Using Three iESBs in Three Factories

In order to evaluate the conceptual architecture, let us run a pencil and paper example of production ramp-up of a new airplane. This is based on a use case [27] within the context of the ARUM project *http://www.arum-project.eu*. The scenario consists of three factories (see Fig. 3) where an iESB runs in each of them but instantiated differently from each other: Factory A employs software agents as Intelligent Services, factory B uses Web services and a set of legacy systems connected through gateways. Factory C runs a combination of agents and Web services. All factories have to communicate with each other in order to synchronise and optimise a production cycle, avoid delays and so on.

Let us assume that the factories receive instructions for producing the first aircraft of a new series. The production is distributed among the factories using the Process Manager and each stage has a clear deadline. The whole plan is arranged in such a way to meet the last deadline is when the aircraft has to be delivered.

First, an initial scheduling of the production is performed. The operations are allocated to the stations and workers at a factory level according to the specified production processes. A Scheduler in factory A identifies that it does not have enough resources to fulfil the scheduled operations in time. The notification is sent via ESB to other factories, whose connection have been established in advance via the Node Manager. Replanning and rescheduling are then initiated. As a result of negotiations involving all factories using the Peer-2-peer Layer, several operations originally assigned to factory A are moved to factory B. The stations and workers receive their new schedules and confirm them. After the distributed process is finalised and all steps are distributed, the production begins.

Factory A executes its part of the work and passes a part of the aircraft to factory B. Factory B starts the execution of the assigned operations but soon enough one of its Web services is automatically notified through the ESB about an emerging issue. An inventory/warehouse connected via Gateway Service push a message to the ESB and a Scheduler notices that the required supplies will not be delivered in time. After getting a new delivery date from a supplier, the Scheduler in factory B raises a flag via ESB to the whole network and communicates a delay in operation and the approximate date for finishing its allocated task, taking into account the new delivery date. This chain of events triggers rescheduling for the whole production plan because the order will not be fulfilled in time. Thus the dependant tasks also have to be delayed and rescheduled and so on. However, the agent-based Web service Planner at factory C has discovered that there is available stock of the required supplies in its warehouse and offers to send it to factory B. The rescheduling is initiated again based on these new facts.

In order to reschedule across several factories, the Schedulers in all factories negotiate through the Peer-2-peer Layer Service allowing the Schedulers to talk to each other effectively and using all the facilities needed for one-to-one and many-to-many negotiations. Notice that the factories always try to find the best solution to eliminate the consequences of unexpected events within their own environment by trying to adjust their own schedules and use their own resources. If they do not succeed, they interact with other participants of the process via ESB to work out a mutual solution.

According to the recent changes the order will be fulfilled within the deadline. The stations and workers receive a new schedule and continue their respective operations. Bear in mind that the Scheduler and Planner Services are constantly operating, trying to find the optimal solution all the time. If the schedule horizon has not passed the point when it would be too late to change a working order it is always possible to commit changes and improve the future steps.

Intelligent Services do not relay on a particular implementation technology as long as such Services produce their expected result. Thanks to the ESB interoperability issues between technologies are minimised while enabling rapid responses by connected Services.

VI. Discussion

In the running example, three instances of the conceptual architecture are used. Each instance uses a different technology, yet they all together manage to deal with the unexpected events in the manufacturing process. Notice the advantage of using the ESB as a transparent message delivery, regardless of who is communicating with whom or what technology is used for implementing Intelligent Services. The idea of Intelligent Services without committing to a specific technology allows us to see architecture components in a more abstract way, focusing on functionality rather than implementation constraints.

Committing a technology, e.g. Web services, the running example would have been more reactive in the sense that Web services are stateless and thus they only respond to calls, lacking proactivity. In such a case, the example would be difficult to run. On the other hand, using only MASs renders the ESB pointless because agents could use their own communication channel. However, interoperability issues arise when agents do not follow a common communication protocol. Combining agents with Web services provides some flexibility and minimises interoperability issues. Nevertheless, it all depends on how such combination is carried out, which might impose a combination "flavour" on the overall system, thus restricting its functionality. Our approach, using the ESB, allows freedom of implementing technology, consequently benefiting from them all.

In this paper we focus on small lot production manufacturing as a domain of our approach. Without a substantial implementation we can simply speculate that, in principle, our approach can be applied to other domains with similar requirements such as those presented in Sect. III, cf. business ecosystems [28].

VII. Conclusions

This paper discusses an innovative conceptual architecture for the support of small lot ramp-up and production manufacturing, based on the new definition of Intelligent Services. In this work, Intelligent Services should not focus on implementation details but rather on the general functionality, improving the planning and scheduling activities. As a result, Intelligent Services can be instantiated by intelligent agents, Web services or combinations of them. An intelligent enterprise service bus (iESB) is used as backbone for supporting the interoperability among all these intelligent services provided by different tools.

At the moment, the proposed approach, developed under the project ARUM, is being conceptually developed. Future work will be devoted to the development of the iESB and associated tools, and testing for future evaluations, in particular, performance comparisons with implementations where all components are agents, Web services and their combinations.

Acknowledgment

This work was supported by the European Commission FP7 Programme under the project ARUM No. 314056.

References

[1] N. Mehandjiev and P. Grefen, *Dynamic business process formation for instant virtual enterprises*. Springer, 2010.

[2] P. Verstraete, B. S. Germain, P. Valckenaers, H. V. Brussel, J. V. Belle, and K. Hadeli, "Engineering manufacturing control systems using PROSA and delegate MAS," *International Journal of Agent-Oriented Software Engineering*, vol. 2, no. 1, pp. 62–89, 2008.

[3] W. Shen, Y. Li, Q. Hao, S. Wang, and H. Ghenniwa, "Implementing collaborative manufacturing with intelligent Web services," in *The Fifth International Conference on Computer and Information Technology, CIT 2005.*, pp. 1063–1069, IEEE Computer Society, sep 2005.

[4] P. Leitão, V. Mařík, and P. Vrba, "Past, Present, and Future of Industrial Agent," *IEEE Transactions on Industrial Informatics*, 2013.

[5] M. Wooldridge, *An Introduction to Multi-Agent Systems*. John Wiley & Sons, 2002.

[6] S. Bussmann and K. Schild, "An agent-based approach to the control of flexible production systems," in *8th IEEE International Conference on Emerging Technology and Factory Automation*, vol. 2, pp. 481–488, IEEE, 2001.

[7] P. Vrba, P. Tichý, V. Mařík, K. Hall, R. Staron, F. Maturana, and P. Kadera, "Rockwell automation's holonic and multi-agent control systems compendium," *IEEE Transactions on System, Man and Cybernetics, C: Appications Review*, vol. 41, pp. 14–30, Jan 2011.

[8] FIPA, "Abstract Architecture Specification. Standard of the Foundation for Intelligent Physical Agents," tech. rep., Foundation for Intelligent Physical Agents, 2002.

[9] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. Wiley, 2007.

[10] F. Jammes, H. Smit, J. L. M. Lastra, and I. Delamer, "Orchestration of Service-Oriented Manufacturing Processes," in *10th IEEE International Conference on Emerging Technologies and Factory Automation*, vol. 1, pp. 617–624, IEEE, 2005.

[11] W.-J. V. D. Heuvel and Z. Maamar, "Moving Toward a Framework to Compose Intelligent Web Services," *Commununications of the ACM*, vol. 46, pp. 103–109, Oct. 2003.

[12] L. Ribeiro, J. Barata, and P. Mendes, "MAS and SOA: Complementary Automation Paradigms," in *Innovation in Manufacturing Networks*, vol. 266 of *IFIP – The International Federation for Information Processing*, pp. 259–268, Springer US, 2008.

[13] OASIS, "Reference Model for Service Oriented Architecture. OASIS Standard," tech. rep., OASIS, Oct 2006.

[14] J. Mendes, P. Leitão, F. Restivo, and A. Colombo, "Service-oriented Agents for Collaborative Industrial Automation and Production Systems," in *4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS'09)*, vol. 5696 of *Lecture Notes in Artificial Intelligence*, pp. 1–12, Springer, 2009.

[15] S. Jacobi, C. Hahn, and D. Raber, "Integration of Multiagent Systems and Service Oriented Architectures in the Steel Industry," in *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'10)*, vol. 2, pp. 479–482, IEEE, 2010.

[16] H. Fayçal, D. Habiba, and M. Hakima, "Integrating Legacy Systems in a SOA using an Agent based Approach for Information System Agility," in *International Conference on Machine and Web Intelligence (ICMWI'10)*, pp. 338–343, IEEE, 2010.

[17] R. L. Storch and S. Lim, "Improving Flow to Achieve Lean Manufacturing in Shipbuilding," *Production Planning and Control*, vol. 10, no. 2, pp. 127–137, 1999.

[18] G. Heike, M. Ramulu, E. Sorenson, P. Shanahan, and K. Moinzadeh, "Mixed Model Assembly Alternatives for Low-volume Manufacturing: The Case of the Aerospace Industry," *International Journal of Production Economics*, vol. 72, pp. 102–120, 2001.

[19] D. Noack and O. Rose, "A Simulation Based Optimization Algorithm for Slack Reduction and Workforce Scheduling," in *Proceedings of the 2008 Winter Simulation Conference*, pp. 1989–1994, IEEE, 2008.

[20] F. Mas, J. Rios, J. L. Menendez, and A. Gomez, "A Process-oriented Approach to Modeling the Conceptual design of Aircraft Assembly Lines," *International Journal of Advanced Manufacturing Technology*, p. accepted for publication, 2012.

[21] M. F. Majohr, *Heuristik zur personalorientierten Steuerung komplexer Montageprozesse*. PhD thesis, Dresden University of Technology, 2008.

[22] J. Rios, F. Mas, and J. L. Menendez, "Aircraft Final Assembly Line Balancing and Workload Smoothing: A Methodical Analysis," *Key Engineering Materials*, vol. 502, pp. 19–24, 2012.

[23] T. Horenburg, J. Wimmer, and W. A. Günthner, "Resource Allocation in Construction Scheduling based on Multi- Agent Negotiation," in *Proceedings 14th International Conference on Computing in Civil and Building Engineering*, 2012.

[24] G. Ziyaeva, E. Choi, and D. Min, "Content-Based Intelligent Routing and Message Processing in Enterprise Service Bus," in *International Conference on Convergence and Hybrid Information Technology (ICHIT '08)*, pp. 245–249, IEEE, 2008.

[25] A. Goryachev, S. Kozhevnikov, E. Kolbova, O. Kuznetsov, E. Simonova, P. Skobelev, A. Tsarev, and Y. Shepilov, "Smart Factory: Intelligent System for Workshop Resource Allocation, Scheduling, Optimization and Controlling in Real Time," in *Proceedings of the 2012 International Conference on Manufacturing*, vol. 630 of *Advanced Materials Research*, pp. 508–513, Switzerland: Trans Tech Publications, nov 2013.

[26] U. Inden, N. Mehandjiev, L. Mönch, and P. Vrba, "Towards an Ontology for Small Series Production," in *6th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS'13)*, Lecture Notes in Artificial Intelligence, Springer, 2013.

[27] EADS, "Use-Case Definition (use cases 1 & 2)," tech. rep., ARUM Consortium, Aug 2013.

[28] C. A. Marín, I. D. Stalker, and N. Mehandjiev, "Engineering Business Ecosystems Using Environment-Mediated Interactions," in *International Workshop on Engineering Environment-Mediated Multi-Agent Systems*, vol. 5049 of *Lecture Notes in Computer Science*, pp. 240–258, Springer, 2007.