

High Availability Cloud Manufacturing System Integrating Distributed MES Agents

Silviu Răileanu, Florin Anton and Theodor Borangiu

Abstract The paper describes a semi-heterarchical manufacturing control solution based on a private cloud infrastructure which collects data in real time from intelligent devices associated to shop-floor entities: resources and mobile devices embedding the work in process (WIP) on products during their execution cycle. The cloud platform acts as a centralized system scheduler (SS), planning jobs and allocating resources optimally at batch level, and integrates real-time data and status information from agentified shop floor devices. The cloud infrastructure is also used for storage of historic data, for manufacturing set up and control (configuring resource teams for production orders received, selecting the control strategy choice, dynamic rescheduling of order execution in case of resource failure or rush order occurrence) and for hosting the web interface for: remote cell monitoring, reception of client requests, cell configuration, raw material inventory and reports generation. Implementation and experimental results are reported.

Keywords Cloud manufacturing • Private cloud • MES • High availability • Distributing intelligence • Mobile/intelligent devices • Agentification

1 Introduction

Control paradigms for the manufacturing domain have evolved over time from centralized to decentralized or semi-heterarchical [1] and were mainly driven by the new trends in information and communication technology (ICT) such as: mobility,

S. Răileanu (✉) · F. Anton · T. Borangiu

Department of Automatic Control and Applied Informatics, University Politehnica of Bucharest, Bucharest, Romania
e-mail: silviu.raileanu@cimr.pub.ro

F. Anton
e-mail: florin.anton@cimr.pub.ro

T. Borangiu
e-mail: theodor.borangiu@cimr.pub.ro

connectivity, increase of the decisional capabilities, service orientation and more recently the usage of cloud infrastructures to host control applications, run intensive optimization procedures and store large amount of production and resource data [2]. This is how the concept of cloud manufacturing (CMfg) arose [3]; its scope is to handle manufacturing resources and processes according to the quality of services (QoS) they provide, and thus efficiently operate and manage them in a unified manner, providing high availability and flexibility in realizing customer orders in both small batches and mass production [2].

At the manufacturing execution system (MES) level, cloud computing refers mainly to virtualization of applications such as mixed batch planning, job scheduling and resource allocation, or product traceability and production tracking [4]. While MES implementations are different and dependent of the physical infrastructure, the generic MES functions are standardized by ISA-95.03 specifications [5]. ISA-95 defines 5 levels for the hierarchical organization of a manufacturing enterprise, as follows: (i) layers 0, 1 and 2 represent the process control level together with its associated intelligent devices and control system, (ii) layer 3 represents the manufacturing operating level and consists of a series of activities such as scheduling, quality management, maintenance, production tracking, a.o. and (iii) layer 4 is in charge with managing the business-related activities of the manufacturing operation.

In conventional production structures with multiple workstations which allow job-shop fabrication scenarios, there are several physical control and computing entities that are single points of failure (SPoF) which can be avoided only by hardware redundancy; among these entities which execute centralized tasks are the system scheduler (for high performance computing (HPC) tasks), the product router (the controller (PLC) responsible for routing products throughout the cell towards assigned resources), and the central application for production tracking and resource monitoring.

Private cloud platforms integrated as IaaS (Infrastructure as a Service) in manufacturing solve the high availability (HA) problem through resource virtualization techniques and provisioning of computing resources (CPU, storage, I/O) and applications. Also, the system scheduler (SS) implemented in the cloud infrastructure must communicate with the distributed MES (dMES), in which manufacturing resources and product execution orders are agentified; these agents cooperate in a multi-agent framework (MAS), both with the SS to receive scheduled orders and report their execution, and between them at dMES level [6].

Secure communication protocols must be used in such CMfg systems in order to access real-time production data and resource status information from agentified devices: (i) mobile intelligent embedded devices containing both the order execution data (scheduled operations for product execution and assigned resource for each operation) and the WIP data over the product's execution lifecycle; (ii) stationary computing devices providing resource status data and receiving from the SS application programs for assigned operations on products [7, 8].

The paper is structured in six sections: Sect. 2 describes the structure of the dual SS-dMES control architecture, Sect. 3 describes the dynamic model of the control

architecture, Sect. 4 describes the cloud infrastructure which will host the centralized system scheduler and integrating architecture, Sect. 5 presents an experimental case study and the last section is devoted to conclusions.

2 The Control Architecture

The proposed semi-heterarchical control architecture performs manufacturing tasks as follows: (i) resource team configuring, batch planning, product scheduling, resource allocation, cell and production monitoring will be done on the upper SS level running in cloud infrastructure for the received batch orders, and (ii) execution and rescheduling of orders in execution will be done on the lower dMES level; information will be communicated to the upper level for recording and decision making at special event occurrence Resource breakdown/recovery, rush order, local part storage depletion. The choice of having centralized batch optimization and decentralized production control (which may override the centralized optimization) is justified by the HPC availability to run complex optimization programs on far (batch) horizon while having the ability to quickly react at unexpected events possible through intelligence distribution, agentification and decentralization of the control structure. The advantage of this semi-heterarchical control architecture is that it can deal rapidly and locally with orders in execution while computing in parallel at the upper level with high availability an optimized schedule for the orders waiting to be executed. This reduces the myopia of the system at global batch level and preserves the system's agility.

The semi-heterarchical control solution is designed around a common database containing four types of information distributed in 10 tables: (i) shop-floor level information (1—available resources, 2—available operations, 3—products that can be executed, 4—distribution of operations on resources, 5—operations needed to execute products), (ii) clients information (6—clients identification and 7—placed orders), (iii) information resulted as a consequence of the planning and scheduling (8—sequence of orders, sequence of operations, the resources processing the operations and the specific time duration), and (iv) information needed to communicate between the web interface and the shop-floor level (9—operation execution reports and 10—synchronization data). The composing entities are instrumented and the categories of information above defined are updated and processed as depicted in Fig. 1.

- (A) Orders, represented by WIP data, are instrumented with intelligent embedded devices capable to run a multi-agent system and communicating over wireless networks. From an informational point of view the WIP is represented by an order agent. Each order agent sends to the database located on cloud the order and operations status. It receives from cloud the product type, the operations to execute according to a certain control strategy (hierarchical/heterarchical).

3 The dynamic

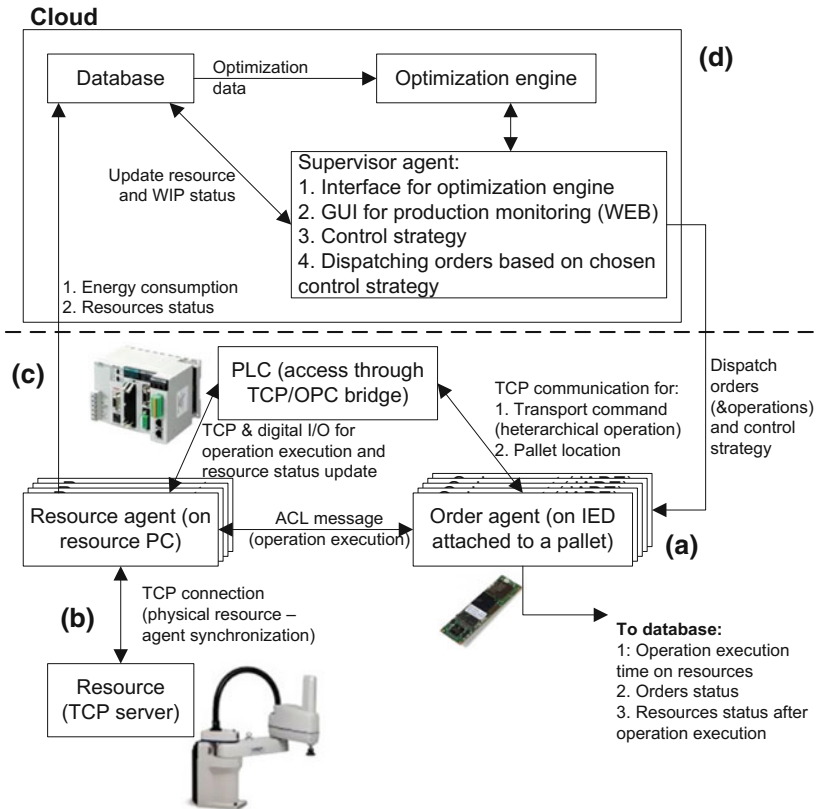


Fig. 1 The 2-layer generic model for manufacturing control

(B) Processing resources (robots, CNC machines) have computing terminals (PCs) attached to their controllers. The PCs run a multi-agent system used to integrate the resources with the mobile embedded devices (located on pallet carriers) embedding WIP, and with the cloud system. From an informational point of view the resources are represented by resource agents who run on the attached PC and send the periodically updated resource status to the database located on cloud.

Resource and order agents are located on the same network and exchange execution information (search operations on resources, request execution of an operation, receive acknowledges from resources).

(C) Transporting resources are represented by the devices of the conveyor (track motors, lifts, diverting elements, etc.) controlled by a PLC which is accessed through a TCP/OPC bridge. This type of resource communicates *only* with the mobile order agents uploaded on the embedded devices of each pallet where a

product will be progressively executed by the visited resources, and fulfils transport services (ex.: order agent requests transport to a specified resource, transport resource realizes the transport operation and an acknowledgement is issued upon completion).

- (D) The cloud infrastructure hosts a set of machines running the database, the optimization engine, the web server and the agent used to synchronize the data from the database with the optimization engine and with the shop-floor level.

3 The Dynamic Model

In order to provide coupling between applications running on the cloud infrastructure and shop-floor devices/resources composing the dMES [9], as well as a flexible message structure which helps the integration of composing entities, the Java Agent Development Framework (JADE) [10] has been used. JADE is specially designed for the development of decentralized systems; control systems also fall into this category, where software agents are loosely coupled and communicate through messages standardized by the Foundation for Intelligent Physical Agents (FIPA) (<http://www.fipa.org>).

Thus, each of the composing entities of the control architecture presented in Fig. 1 has an associated agent: WIP is represented by an **order agent**, the resources (both processing and transporting) are represented by a **resource agent** and a **supervisory agent** runs on the cloud infrastructure. The result is an event-driven architecture (EDA) [11] composed of individual agents used in a MAS framework for monitoring and control; EAD's main advantages are: easy integration of new entities, smart data utilization and an asynchronous and non-blocking operating mode.

The operating process (Fig. 2) of the control system is divided into several sub-processes as follows:

- *Continuous update of the GUI* with information from shop-floor entities (resources and WIP): resource confirm their online status periodically (each minute) and order agents update the state of operations associated to resources after their execution (Dynamic data for interface creation; System state and WIP, Update WIP and visited resources state, Confirm the realization of a product, Update resource data).
- *Receipt of orders* (orders are inserted through the CMfg GUI on the cloud platform and stored in the associate database segment).
- *Production optimization* (orders are planned and scheduled taking into account the resource availability); an optimization algorithm is selected and configured by the user through the cloud GUI according to the enterprise strategy.
- *Set up of the team of resources* that will be involved in the execution of the batch and updating their availability status.

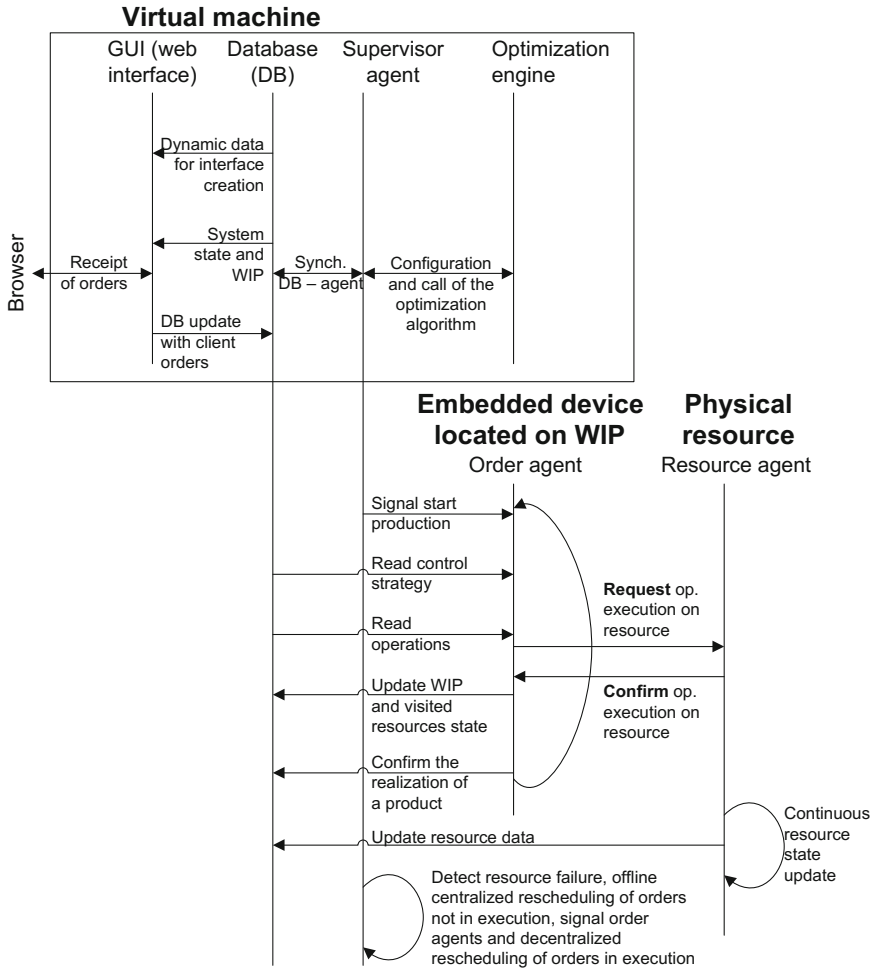


Fig. 2 The operating process of the control architecture

- *Transmission of planned and scheduled orders* to order agents embedded on mobile devices entering the shop floor for product execution. The supervisor agent is in charge with the individual transmission of each planned and scheduled order to a dedicated order agent running on the embedded devices entering the cell. (Signal start production, Read control strategy, Read operations).
- *Execution*: each order agent will be in charge with the execution of its associated product; when the product is finished, the pallet carrying the final product exists the cell and the order holon is deleted, allowing thus the cloud SS to transfer a new order agent for the execution of the next planned product.

- *Detection of resource failure and storage depletion.* These events are discovered due to continuous resource state update and monitoring of production execution; they generate a rescheduling of orders on the newly available resources. Resource update is done by resource agents in order to confirm they are online; they are sent periodically (each minute) to the cloud infrastructure. Execution monitoring is done by order agents, which are asynchronously tracking the progressive realization of operations on resources.

4 HA Design of the Cloud Infrastructure

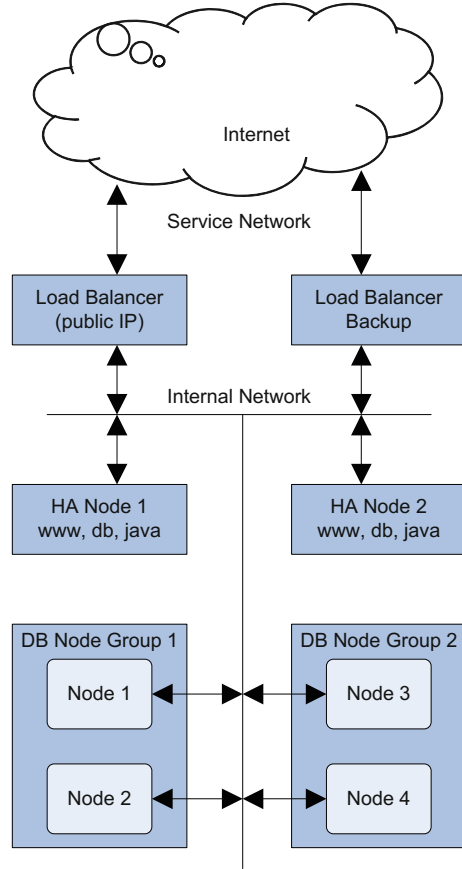
The upper control layer (Fig. 1) has been implemented using an IaaS (Infrastructure as a Service) cloud system running the supervisor agent, database and optimization engine. The cloud system used is an ISDM (IBM Service Delivery Manager) based on Tivoli Service Automation Manager (TSAM) version 7.2 which uses as a hypervisor VMware ESXi 4.1 [12]. Because the applications are running in cloud one could consider that the applications benefit from all the cloud facilities, such as High Availability (HA) offered by the hypervisor and also Distributed Resource Sharing (DRS) offered by the hypervisor too. In reality the situation is not as simple as that; using a cloud system does not mean that the CMfg applications will just inherit the same facilities like the platform they are using.

Because the cloud system is an IaaS, it offers virtual machines as a service; these virtual machines can be used to set up the environment in order to run the applications: in our case a complex database updated with differentiated timing and the supervisor agent. The services in cloud are offered with the facilities that the cloud system supports like HA, meaning that the virtual machines (VM) used will be always available, but that does not guarantee that the applications inside the VM will be offered with HA. The reason for this is that the software could have bugs (the applications or the operating system) which will render the CMfg service unavailable, even if the VM is properly running from hardware point of view.

In order to solve this high availability problem extended to real time CMfg monitoring and control applications, an HA cluster was foreseen for database and application availability. The solution we propose is depicted in Fig. 3 and is composed by the following VMs:

1. Two load balancers (VMs) which are running in a cluster; the *Load Balancer VM* is publicly available and can be accessed from Internet. The function of the Load Balancer cluster is to get requests from internet for two types of services: secure https requests (port 443) and JAVA agent communication, and to forward these requests to the HA cluster in the internal network. The Load Balancer acts also as a gateway for the HA Nodes. For the JAVA agent communication requests a VPN tunnel implemented with IPSec (Internet Protocol Security) [13] has been created between the Load Balancer and the manufacturing infrastructure. In this way the https requests are coming from Internet and the JAVA

Fig. 3 The VM infrastructure in cloud



agent communication requests are coming from an internal network over an encrypted communication channel using VPN. The requests are processed by the Load Balancer and distributed to the HA Nodes using a Round Robin algorithm in order to load evenly the nodes. The HA Nodes are checked for service availability by many agents (agents used to verify if the Node is up and running and the service is accessible) from the Load Balancer. If a service is unavailable on a HA Node the session will be commuted to the other node and the other incoming requests are sent to the available node until the failed one recovers. The *Load Balancer Backup* is used only to check the status of the main Load Balancer; if the main Load Balancer fails, the backup one will take over by assigning the public IP and starting the load balancing function—in this way it becomes the main load balancer and the failed node becomes the backup one. The load balancer is implemented using the piranha [14] load balancer from RedHat on the Linux operating system.

2. The HA Cluster is composed by two nodes (VMs) and offers availability for three services: (i) *web interface access* using the https protocol on port 443; the service uses apache as a web server and the connection is encrypted using 1024 bit SSL (Secure Socket Layer) certificates [15]; (ii) *JAVA agent communication*, the connections being made between the agent which is running on the HA node and the agents on the manufacturing infrastructure; the connection is encrypted in Internet (an IPSec tunnel has been created between the Load Balancer and the gateway of the manufacturing infrastructure), the connection is unencrypted in the internal network; (iii) *database access*: the database management system used is a MySQL database implemented using MySQL Cluster for High Availability.

The HA Cluster is configured to execute the services on both nodes in parallel which is quite unusual for the HA solution which was chosen, namely a RedHat High Availability Cluster. This configuration has been designed in order to offer a plus of performance by using load balancing. The behaviour of the cluster if a service fails is to try to restart that service on the node where the service failed; if this action has no effect and the service is already running on the other node the service will be stopped on the current node. The last special situation is when the service is not running on both nodes and the restart has failed; in this situation the service will be migrated on the other node. The fencing function (a function which isolates the failed node in order to avoid data corruption) was inhibited on the cluster because the two nodes can execute the services in parallel without restrictions.

3. The MySQL cluster is using four VMs grouped in two Node Groups. The Database is distributed between the two Node Groups, for example if we consider a table on the database which has four fields (F1, F2, F3 and F4), Node 1 will store F1 and F3, Node 2 will store F3 and F1, Node 3 will store F2 and F4, and Node 4 will store F4 and F2. Between the nodes the cluster offers data replication and data consistency. In the example above the database will be available and consistent even if two nodes from two node groups will fail. The MySQL Cluster protects against outages offering the following facilities:

- *Synchronous Replication*—Data within each data node is synchronously replicated to another data node.
- *Automatic Failover*—the heart beating mechanism detects in real time any failures and automatically fails over to other nodes in the cluster, without service interruption.
- *Self-Healing*—Failed nodes are able to self-heal by automatically restarting and resynchronizing with other nodes before re-joining the cluster, with complete application transparency.
- *Shared Nothing Architecture, No Single Point of Failure*—each node has its own disk and memory, so the risk of a failure caused by shared components such as storage, is eliminated.

If an error event is detected on any level of the infrastructure (Load Balancer, HA Cluster or MySQL Cluster) a notification to the system administrator will be sent in order to inform about the problem and to offer the opportunity to debug and solve the malfunctions.

5 Case Study: Experimental Results

Experiments have been done in order to validate data collection from shop-floor entities (consisting of mobile devices associated to WIP and resources) and the integration of this data into a cloud infrastructure. The results are shown in Fig. 4.

Thus, a batch of orders has been defined and launched in execution using the control application running on cloud. The steps taken are: update resource status, define product recipes using the available operations, define batch of orders from the available products, plan and schedule orders, validate the delivery date based on the computed makespan, select semi-heterarchical as control strategy, launch in execution planned and scheduled orders.

During the execution period a resource has changed its state from online to offline, no longer being available for production (resource failure). Since it no longer updates its status in the cloud database it will be considered offline and the remaining batch of orders was rescheduled using the available resources while the orders in execution (WIP) are negotiating in real-time the allocation of remaining operations on the available resources.

Since the data synchronization between the two levels (cloud and shop-floor devices) is practically instantaneous, the efficiency of using the cloud infrastructure

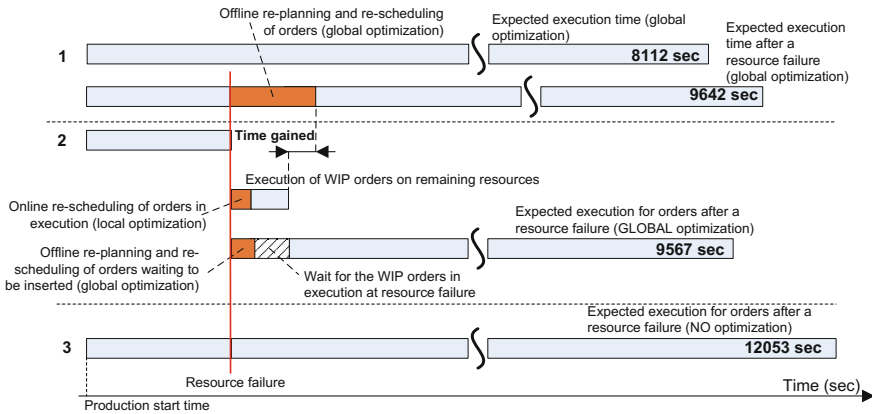


Fig. 4 Comparison of the execution times for the 3 different execution strategies: 1 centralized planning and scheduling (done in cloud) for all orders, 2 centralized planning and scheduling for orders waiting execution and decentralized re-scheduling for WIP orders, 3 decentralized planning and scheduling for all orders

to realize the offline planning and scheduling consists in the fact that production is still running with local, order agents optimization at shop-floor level until the Cloud re-planning and rescheduling of remaining products are performed and the hierarchical mode is resumed. Thus, time is gained by realizing the two activities in parallel (Fig. 4), since the Cloud SS optimization of scheduling a large batch of orders takes less time (due to its speed HPC characteristics) in comparison with classical SS implementing, thus guaranteeing the termination of rescheduling until the execution of WIP completes.

6 Conclusions

The paper describes a semi-heterarchical manufacturing control solution based on a private cloud infrastructure which collects data in real time from intelligent devices associated to shop-floor entities: resources and mobile devices embedding the WIP on products during their execution cycle. The cloud platform acts as a centralized SS, planning jobs and allocating resources optimally at batch level, and integrates real-time data and status information from agentified shop floor devices. The advantages brought by this control architecture are: easy integration of manufacturing data into a cloud infrastructure using a multi-agent framework distributed on the field entities and on the cloud (Fig. 1), fault tolerance and high availability of the applications centralized at MES level, running on cloud (Fig. 3) and the possibility to run in real manufacturing time intensive applications (such as optimization engines used for re-planning and re-scheduling in case of shop floor disturbances) in the cloud (Fig. 3).

A Public Key Infrastructure (PKI) solution with SSL authentication and encryption for intelligent products travelling on pallets in the shop floor and embedding WIP data to be transferred to the Cloud has been designed. From an implementation perspective, SOA alignment at shop floor level involves TCP/IP-based communication over Wi-Fi supporting higher level protocols. Our contribution in the security area CMfg implementation of the communication between the Cloud SS and the intelligent devices in the dMES (embedded devices for WIP, resource orders discovering the cell status) considered security challenges associated with data and information flow protection as well as authentication and authorization of the actors involved (order agents and resource agents).

Another advantage of using the Cloud IaaS for manufacturing management is the service oriented connectivity with the upper, business layer of the enterprise, offering direct access of clients to order acceptance, resource availability, and estimation of delivery of products via a cloud GUI. SOA principles can be used for the standardization of the data and information flow from the shop floor (production execution) level to the management (customer order tracking) level of the enterprise.

Regarding the infrastructure and the tools used to implement the CMfg solution, they offer High Availability on two levels: the cloud system offers HA at the

infrastructure level and the Load Balancing/HA Clustering solutions offers HA at the application level.

From the point of view of security, the communication between clients and the web interface is encrypted using self-signed SSL certificates that offer a good level of security by using the Diffie-Helman algorithm [16]. The communication between JAVA agents is also encrypted in Internet by using an IPSec VPN tunnel between the gateways of the manufacturing infrastructure and the load balancer in the cloud infrastructure. The other connections are made in the local network which is a private network implemented with virtual LAN's (VLAN). In the cloud system the HA Cluster and the MySQL cluster are in the same VLAN, but if a higher level of security is required there is the possibility to isolate the MySQL Cluster from the Load Balancer by using a VLAN which will connect the HA Cluster with MySQL Cluster.

Acknowledgements This research work has been partially supported by the IBM FA 2016 Project: Big Data, Analytics and Cloud for Digital Transformation on Manufacturing—DTM.

References

1. Borangiu, Th., Răileanu, S., Berger, T., Trentesaux, D.: Switching mode control strategy in manufacturing execution systems. *Int. J. Prod. Res.* **53**(7), 1950–1963 (2015). doi:[10.1080/00207543.2014.935825](https://doi.org/10.1080/00207543.2014.935825), ISSN: 0020-7543 (Print), 1366-588X (Online)
2. Kubler, S., Holmstrom, J., Främling, K., Turkama, P.: Technological Theory of Cloud Manufacturing, Service Orientation in Holonic and Multi-Agent Manufacturing. *Studies in Computational Intelligence*, vol. 640, pp. 267–276. Springer International Publication (2016). doi:[10.1007/978-3-319-30337-6_24](https://doi.org/10.1007/978-3-319-30337-6_24)
3. Wu, D., Greer, M.J., Rosen, D., Schaefer, D.: Cloud manufacturing: strategic vision and state-of-the-art. *J. Manuf. Syst.* **32**, 564–579 (2013). doi:[10.1016/j.jmsy.2013.04.008](https://doi.org/10.1016/j.jmsy.2013.04.008)
4. Morariu, O., Morariu, C., Borangiu, Th.: Shop-floor resource virtualization layer with private cloud support. *J. Intell. Manuf.* **30**, 1–16 (2014). doi:[10.1007/s10845-014-0878-7](https://doi.org/10.1007/s10845-014-0878-7)
5. Meyer, H., Fuchs, F., Thiel, K.: *Manufacturing Execution Systems*. McGraw-Hill (2009). ISBN: 978-0-07-162602-6
6. Novas, J.M., Bahtiar, R., Van Belle, J., Valckenaers, P.: An approach for the integration of a scheduling system and a multi-agent manufacturing execution system. towards a collaborative framework. In: *Proceedings of the 14th IFAC Symposium INCOM'12*, Bucharest, IFAC PapersOnLine, pp. 728–733 (2012)
7. Morariu, O., Morariu, C., Borangiu, Th.: Security issues in service oriented manufacturing architectures with distributed intelligence. In: Borangiu, T., Trentesaux, D., Thomas, A., McFarlane, D. (eds.) *Service Orientation in Holonic and Multi-Agent Manufacturing*. Springer Series SCI, vol. 640, Chapter 5.4, pp. 205–224. Springer International Publication (2016). doi:[10.1007/978-3-319-30337-6_23](https://doi.org/10.1007/978-3-319-30337-6_23), ISBN: 978-3-319-30335-2
8. Wang, L., Shen, W., Lang, S.: Wise-ShopFloor: a web-based and sensor-driven shop floor environment. In: *The 7th International Conference on IEEE Computer Supported Cooperative Work in Design*, pp. 413–418 (2002)
9. Främling, K., Kubler, S., Buda, A.: Universal messaging standards for the IoT from a lifecycle management perspective. *IEEE Internet Things J.* **1**(4), 319–327 (2014)
10. Bellifemine, F., Carie, G., Greenwood, D.: *Developing Multi-agent Systems with JADE*. Wiley (2007). ISBN: 978-0-470-05747-6

11. Theorin, A., Bengtsson, K., Provost, J., Lieder, M., Johnsson, C., Lundholm, T., Lennartson, B.: An event-driven manufacturing information system architecture for Industry 4.0. *Int. J. Prod. Res.* (2016). doi:[10.1080/00207543.2016.1201604](https://doi.org/10.1080/00207543.2016.1201604)
12. Buyya, R., Vecchiola, C., Selvi, T.: *Mastering Cloud Computing Chapter 3—Virtualization*, pp. 71–109 (2013). doi:[10.1016/B978-0-12-411454-8.00003-6](https://doi.org/10.1016/B978-0-12-411454-8.00003-6)
13. Rowan, T.: VPN technology: IPSEC vs SSL. *Netw. Secur.* **2007**(12), 13–17 (2007)
14. Red Hat Enterprise Linux 5: *Virtual Server Administration*, Red Hat (2009)
15. Meyer, H.: Securing intranet data with SSL client certificates. *Comput. Secur.* **16**(4), 334 (1997)
16. Harn, L., Lin, C.: Efficient group Diffie-Hellman key agreement protocols. *Comput. Electr. Eng.* **40**(6), 1972–1980 (2014)

Service Orientation in Holonic and Multi-Agent
Manufacturing

Proceedings of SOHOMA 2016

Borangiu, T.; Trentesaux, D.; Thomas, A.; Leitão, P.;
Barata Oliveira, J.A. (Eds.)

2017, IX, 438 p. 136 illus., Hardcover

ISBN: 978-3-319-51099-6