Evaluating Service-Oriented Orchestration Schemes for Controlling Pallet Flow

Johannes Minor, Jorge Garcia, Jaacan Martinez, Andrei Lobov, Jose L. Martinez Lastra
Tampere University of Technology
Tampere, Finland
{johannes.minor, jorge.garcia, jaacan.martinez}@tut.fi, {lobov, lastra}@ieee.org

***Abstract*—Incorporating web services at the device level is expected to improve many aspects of automated manufacturing systems, including scalability, reusability, reconfigurability, compatibility between equipment from different vendors, and cross-layer integration. There is much ongoing research into architectural issues and enabling technologies for service-oriented automation, but the body of knowledge surrounding service identification and optimal orchestration schemes needs improvement. In this project, the performance of a test system, consisting of conveyors with embedded web service-enabled devices, with three different orchestration schemes is evaluated. The three schemes have similar message exchange patterns, but differ in terms of the degree of involvement of a central orchestrator. The schemes are compared in terms of pallet transfer timing, and communications load. Performance is similar for the small test system, but the results predicted scalability problems for the schemes with high reliance on a central orchestrator.**

Keywords-SOA; orchestration; control schemes.

I. INTRODUCTION

The case for applying Service Oriented Architecture (SOA) to industrial control systems has been made in many previous research projects [1,5,10]. The benefits that Service Orientation at the device level is expected to bring to industrial applications include increased business agility, easier and less costly equipment reusability and reconfigurability, and improved cross-layer integration. In addition, building systems around open standards reduces a business' reliance on proprietary protocols, systems, and data formats, and can drastically decrease the effort with which systems from multiple vendors can be integrated, allowing factories to choose the best-of-breed components for all systems, without worrying about interoperability.

Much of the research into service orientation in industrial automation has focused on architectural issues, and the enabling technologies and standards. Still missing, however, are the practical implementation details, best practices, and system design methodologies that can help bridge the gap between systems theory and a working SOA deployment that exhibits some or all of the benefits touted by SOA advocates.

Although the benefits of service orientation have been established, additional research is required to expand the body of knowledge surrounding the field. Specific issues that still need to be addressed are:

- How to combine scan-based and event-based systems?
- Down to what level is it reasonable to have web services?
- How best to compose services over a number of distributed devices to execute business processes?
- How to integrate legacy equipment into a SOA-based system?

This paper proposes and evaluates some possible orchestration schemes for handling pallets on a system of conveyors, in an attempt to provide some answers to the following questions:

- What is the best approach for composing atomic services into a more complex task?
- How can we combine event- and scan-based SOA manufacturing systems?

A. Orchestration and Choreography

When discussing SOA in automation, orchestration typically refers to the practice of composing a set of exposed services, with a pre-defined interaction pattern that defines a business or manufacturing process [1]. The process can be described using a language such as Web Services Business Process Execution Language (WS-BPEL). The orchestration engine executes the application logic, sequencing and synchronizing service invocations to reach the business goal [15].

The focus of orchestration is on a high-level view of the process workflow, whereas choreography considers the lower level rules that define the message exchange sequences. The W3C candidate recommendation Web Service Choreography Description Language v1.0 (WS-CDL) [8] can be used to describe peer to peer collaboration by defining their globally observable behavior, where business goals are realized by peer to peer message exchange.

B. Previous Research

EU project SODA [16] investigated the eco-system required to build, deploy, and maintain a SOA application in many domains. The SIRENA [17] project demonstrated the feasibility of extending SOA to the device level, and produced some proof-of-concept device-level services [10]. FP6 SOCRADES [18] evaluated a number of solutions for SOA at the device level, and demonstrated a SOA solution for automating electronics assembly.

Much research has been done on system modeling, control, and decision making at higher levels, with Petri Net-based approaches [2,4] and Timed Net Condition/Event

 ISBN: 978-1-61208-184-7

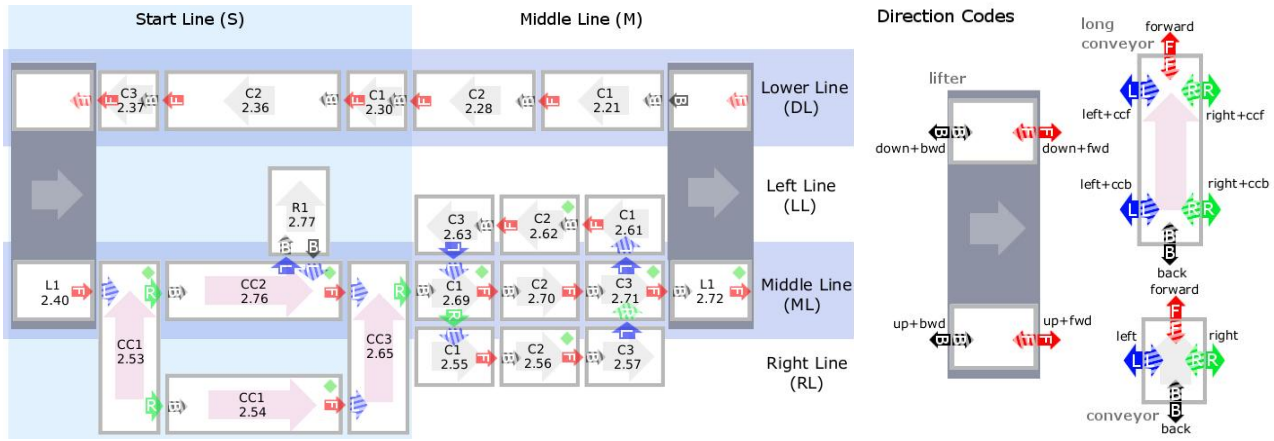

Fig. 1. System of conveyor segments used for testing pallet flow control schemes.

Systems (TNCES) [3]. Other research presents designs for orchestration engines for controlling systems composed of web service-enabled embedded devices [3,11].

Some software engineering approaches to combining, modeling, and optimizing service orchestration and choreography have been proposed [6,7], but the focus has been on eliminating redundant data transfers to minimize process execution time in IT systems. Time-consuming data transfers between devices are not the primary concern on the factory floor.

This paper describes some preliminary research into optimal strategies for composing device-level, fine-granularity, atomic services, to synchronize device interactions to efficiently complete manufacturing processes. This research aims to find a balance between device-to-device interactions, and master-slave control schemes, with long-term goal of providing an easier transition path

Sections II and III introduce the test system, and describe the control schemes. Section IV analyzes the data, and Section V presents conclusions and future work.

II. SYSTEM DESCRIPTION

For this study, three different pallet transfer control schemes are tested using a varying number of pallets on the system shown in Fig. 1.

The system consists of 21 conveyor segments. An intelligent Web Services-enabled device controls each segment. This line uses the InicoTech S1000 Smart RTU [13]. The main loop is made up of the middle and lower lines, connected at either end by a lifter, 13 segments in total. The continuously moving loop acts as a buffer for the robot and manual workstations off the main line. Pallets can be introduced to the system at loading stations in the branches off the main loop.

Each pallet has a unique RFID code, read when a pallet is loaded at a loading station. To transfer a pallet from one conveyor segment to the next, the following message exchange takes place, independent of the control scheme. This pattern is shown in Fig. 2.

1) Reservation Request/Response: A message containing the pallet ID and input transfer direction is sent to the conveyor or lifter. The reservation status, and the ID of the pallet holding the reservation are returned. If the pallet IDs match, and the status is "RESERVED," the transfer can safely proceed. If the device is not in a position to accept a pallet at the requested input position (e.g., reservation is requested at the upper end of a lifter while the lifter is down), the reservation status is "PENDING." Reservation Requests are sent until the reservation is successful.

2) Transfer In Request/Response: A message containing the pallet ID and the input transfer direction are sent. If they match the reservation information, "ACCEPTED" is returned.

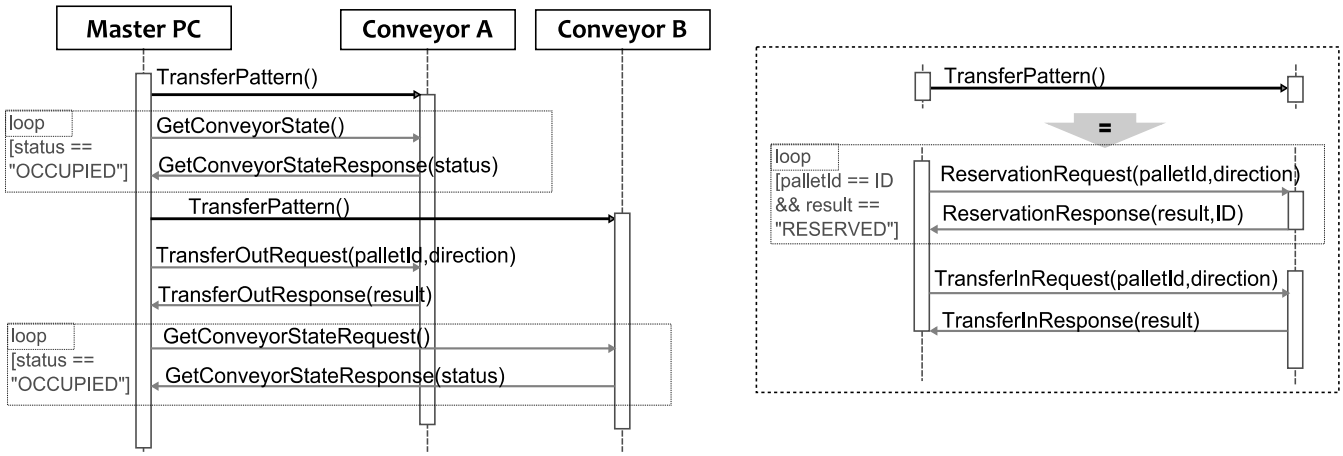

Fig. 2. Master-Slave Control Scheme

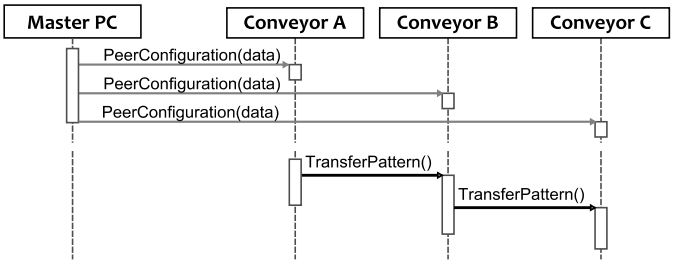

Fig. 3. Peer to Peer Control Scheme, using TransferPattern from Fig. 2.

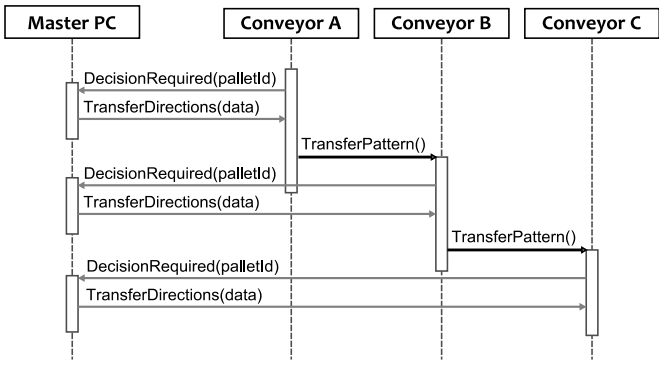

Fig. 4. Combined Control Scheme, using TransferPattern from Fig. 2.

III. CONTROL SCHEME DESCRIPTIONS

For this study, varying numbers of pallets are added to the main loop, driven by different control schemes. The three schemes chosen for comparison represent different balances between device-to-device interaction, and top-down orchestration.

A. Master-Slave: All interactions through Orchestration Engine

Each remote device is a DPWS server, and a single master DPWS client controls the pallet flow by invoking actions on all remote devices. The remote devices themselves do not autonomously invoke actions on other remote devices. Each remote device supports the same set of operations:

1) Reserve: A reservation request message is sent, containing a unique RFID (Radio Frequency IDentification) Code to identify the pallet, and the input transfer direction. Direction codes supported by each device can be determined from metadata in the WSDL read from the device in the discovery phase. This operation returns the reservation state (free, pending, reserved), and the ID of the pallet that the conveyor is reserved for, if any.

2) TransferIn: The transfer-in request message contains the same data as the reservation request. The operation is accepted if the request message matches the reservation data, or rejected if it does not.

3) GetConveyorState: After the *TransferIn* action is invoked, the conveyor state is polled at some interval. The operation is complete when the conveyor state request returns "occupied."

4) TransferOut: When a downstream conveyor is reserved, the transfer-out operation is invoked. The request message contains the pallet Id and the output direction code.

5) ReadRfidTag: Certain devices have an RFID tag reader, which is used to read the unique RFID code of the pallet when it is first loaded, and for consistency-checking during operation.

One process is required on the master PC running the Orchestration Engine for each pallet. The message exchange pattern is shown in Fig. 2. This exchange pattern can be described in WS-BPEL.

B. Peer to Peer: Devices handle pallet transfer autonomously

The devices cooperate autonomously with each other to achieve common goals. Each remote device acts as a combined DPWS client and server, each with the ability to invoke actions on other remote devices. Each device supports the same set of services, and follows the same interaction pattern to allow for collision-free pallet flow. At startup, a supervisory control system dynamically discovers the devices present in the network, and based on some layout map, and invokes a configuration service on each device, containing the following information:

- Output transfer direction code
- Service Address of the downstream peer device
- Input transfer direction to the downstream peer device

When pallet is introduced to the system at a loading station, neighboring devices negotiate pallet transfer with the message exchange pattern shown in Fig. 3, containing the same *Reserve* and *TransferIn* operations as the Orchestrator control scheme. No *TransferOut*, *GetConveyorState*, or *ReadRfidTag* operations are required.

C. Hybrid Approach: Peer to Peer with Decision Request Notifications

In this approach, lower-level pallet transfer control

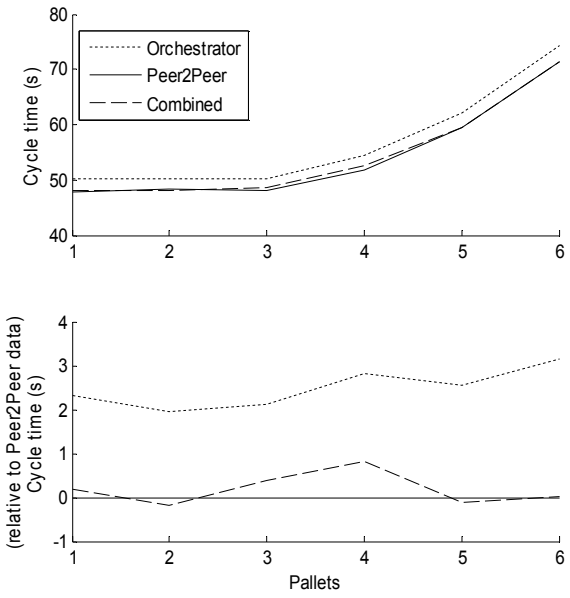

Fig 5. Main Loop Cycle (Lap) Times for different Control Schemes.

messages (*ReservationRequest*, *TransferIn*) are exchanged peer to peer, as in the previous scheme, but the device receives transfer instructions (downstream peer service address, output transfer direction, peer input transfer direction) from the master after the device publishes a notification that a pallet has been received (*DecisionRequest*). This exchange pattern is shown in Fig. 4.

When the system is stared, the master PC dynamically discovers all devices, and subscribes to the *DecisionRequest* notifications. When a notification is received, some logic executed on the master PC determines the appropriate next step, and sends the appropriate transfer instructions.

IV. EXPERIMENT AND RESULTS

The tests were conducted as follows:

1) A configuration file describing the layout of the conveyor system is loaded in the control software running on the master PC. The master PC subscribes to the "*PalletInformation*" notifications on each device, used for logging purposes, common to all control schemes.

2) A single pallet is transferred from conveyor to conveyor around the loop.

3) After fifteen minutes, a new pallet is placed at a loading station, and introduced into the main loop with the existing pallets

4) Additional pallets are introduced at approximately equal intervals for 90 minutes

A. Average Pallet Lap Time

The average time for a pallet to complete one lap of the main loop for the three schemes is shown in Fig 5. The results are plotted relative to the data from the Peer to Peer test, because it had the fastest lap times, on average. These results are not surprising. Although all control schemes use the same interaction pattern for reservation and pallet transfer (i.e. polling at a 500ms interval for reservation), the Master-Slave scheme has an additional, approximately 350ms polling cycle to determine when the TransferIn operation is complete. With thirteen devices in the loop, we would expect the lap time to be approximately 13×(350/2)ms = 2.275s longer for a single pallet.

The performance bottlenecks in the system were the lifters, because of time taken to transfer a pallet between the upper and lower lines. While fewer than four pallets are in the loop, transfer times stay constant, because the pallets do not interfere with each other. For four pallets, the marginally faster performance of each transition using the Peer to Peer approach results in noticeably faster average times for four pallets, but the advantage fades for five or more pallets. This behavior is coincidental, not inherent to the control scheme, and would likely change if the conveyor line layout were changed.

We can conclude that the orchestration scheme chosen has a negligible impact on the timing of pallet transfers.

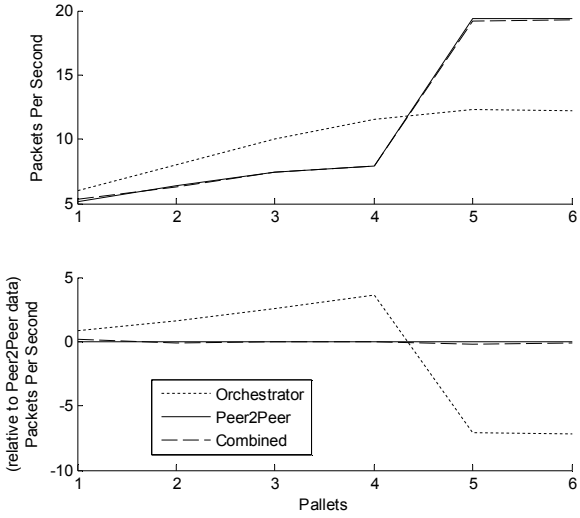

Fig 6. Packets sent per second by a device before the lifter bottleneck

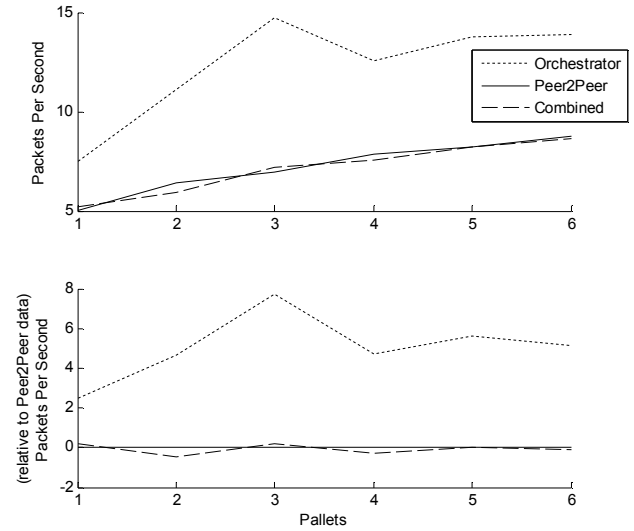

Fig 7. Packets sent per second by a device after the lifter bottleneck

B. Communication Load

Access to network statistics for the devices was limited to the number of packets sent and received since startup. Although this is not useful as an absolute measure, it can be used to compare relative performance. Fig. 8 shows a typical data set for one test for one device. There is a spike in activity when a new pallet is introduced, and then it stabilizes.

Fig. 6 and Fig.7 show the average load for devices before and after the lifter bottleneck. For lower pallet numbers, the communication load is lower devices operating under the peer to peer and hybrid schemes. This is likely because there is no polling taking place. Reservation requests are event-based (i.e. sent when a pallet is received). When the pallets start to interfere with each other, the peer to peer and combined approaches create communication loads almost twice that of the orchestrator approach. However, it is important to note

 ISBN: 978-1-61208-184-7

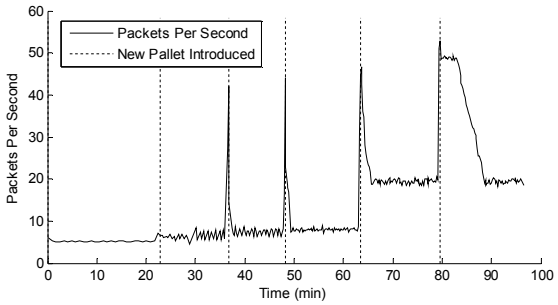

Fig 8. Typical "Packets Per Second" data

that the orchestration engine running on the PC is experiencing a load roughly equivalent to the difference, because the devices are not communicating directly with each other. This raises scalability questions, because the communications load from all devices is concentrated at the orchestrator.

For devices after the bottleneck, as in Fig. 7, communication load remains lower for the event-driven peer-to-peer and combined control strategies.

V. CONCLUSIONS AND FUTURE WORK

This research shows that for small systems, timing of physical operations, such as pallet transfers, are not very sensitive to the choice of orchestration scheme. However, analysis of the communication loads on the remote devices and the orchestrator (or master PC) suggests that performance will degrade with increasing system size for schemes that rely heavily on a central orchestrator. Additional research is required into the scalability of these methods.

This research lays the groundwork for designing and testing more complex systems. A system with autonomous devices interacting requires robust supervisory control and monitoring. Implementing Complex Event Processing (CEP) for monitoring and decision support for orchestration can be also considered. Complex Event Processing, used in conjunction with various formal system modeling methods [2,12], is a promising approach for providing detailed information about the state of the system, as well as fault prediction, prevention, and detection, and deadlock prevention in flow control.

REFERENCES

[1] F. Jammes, H. Smit, J.L. Martinez Lastra, and I.M. Delamer, "Orchestration of service-oriented manufacturing processes," IEEE 10th Conference on Emerging Technologies and Factory Automation (ETFA 05). vol. 1, pp. 617-624, 19-22 Sept. 2005

[2] C. Popescu and J.L. Martinez Lastra, "An incremental Petri Net-derived approach to modeling of flow and resources in service-oriented manufacturing systems," IEEE 8th International Conference on Industrial Informatics (INDIN 10), pp. 253-259, 13-16 July 2010

[3] A. Lobov, J. Puttonen, V.V. Herrera, R. Andiappan, and J.L. Martinez Lastra, "Service oriented architecture in developing of loosely-coupled manufacturing systems," 6th IEEE International Conference on Industrial Informatics (INDIN 08), pp. 791-796, 13-16 July 2008

[4] J.M. Mendes, P. Leitao, A.W. Colombo, and F. Restivo, "Service-oriented process control using High-Level Petri Nets," 6th IEEE International Conference on Industrial Informatics (INDIN 08), pp. 750-755, 13-16 July 2008, doi: 10.1109/INDIN.2008.4618202

[5] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation," IEEE Transactions on Industrial Informatics, vol. 1, no. 1, pp. 62-70, Feb. 2005

[6] B. Haopin, S. Meina, X. Huiyang, W. Qian, and F. Lingyun, "An Optimized Design of Service Orchestration," Third International Conference on Pervasive Computing and Applications (ICPCA 08), vol. 2, pp. 980-984, 6-8 Oct. 2008

[7] J. Sun, Y. Liu, J. Song Dong, G. Pu, and T. Hut Tan, "Model-Based Methods for Linking Web Service Choreography and Orchestration," 17th Asia Pacific Software Engineering Conference (APSEC 10), pp. 166-175, Nov. 30-Dec. 3 2010, doi: 10.1109/APSEC.2010.28

[8] Web Services Choreography Description Language Version 1.0 http://www.w3.org/TR/ws-cdl-10/ (retrieved: January, 2012)

[9] OASIS Web Services Business Process Execution Language V2.0, 11 APR 2007, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html (retrieved: January, 2012)

[10] A.W. Colombo, F. Jammes, H. Smit, R. Harrison, J.L. Martinez Lastra, and I.M. Delamer, "Service-oriented architectures for collaborative automation," 31st Annual Conference of IEEE Industrial Electronics Society (IECON 05), pp. 2649-2654, 6-10 Nov. 2005, doi: 10.1109/IECON.2005.1569325

[11] Y.S. Park, T.D. Kirkham, P. Phaithoonbuathong, and R. Harrison, "Implementing agile and collaborative automation using Web Service orchestration," IEEE International Symposium on Industrial Electronics (ISIE 09), pp. 86-91, 5-8 July 2009, doi: 10.1109/ISIE.2009.5213759

[12] D. Cachapa, R. Harrison, and A.W. Colombo, "Monitoring functions as service composition in a SoA-based industrial environment," 36th Annual Conference on IEEE Industrial Electronics Society (IECON 10), pp. 1353-1358, 7-10 Nov. 2010, doi: 10.1109/IECON.2010.5675485

[13] InicoTech Technologies LTD; S1000 User Manual; http://www.inicotech.com/doc/S1000%20User%20Manual.pdf (retrieved: January, 2012)

[14] J. M. Garcia Izaguirre, A. Lobov, and J.L. Martinez Lastra, "OPC-UA and DPWS Interoperability for Factory Floor Monitoring using Complex Event Processing," 9 th IEEE International Conference on Industrial Informatics (INDIN 11), pp. 205-211, 26-29 July 2011 doi: 10.1109/INDIN.2011.6034874

[15] J. Puttonen, A. Lobov, and J.L. Martinez Lastra, "An application of BPEL for service orchestration in an industrial environment," IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 08), pp. 530-537, 15-18 Sept. 2008

[16] SODA Project Profile, http://www.ims.es/pdf/eng/downloads/publications/SODA_profile_oct-06.pdf (retrieved: January, 2012)

[17] SIRENA Project, http://www.sirena-itea.org/ (retrieved: January, 2012)

[18] SOCRADES Project, http://www.socrades.eu/Home/ (retrieved: January, 2012)

 ISBN: 978-1-61208-184-7