Service-oriented SCADA and MES Supporting Petri nets based Orchestrated Automation Systems

Armando W. Colombo[1,2], J. Marco Mendes[1], Paulo Leitão[3,4], Stamatis Karnouskos[5]
[1] Schneider Electric Automation GmbH, Steinheimer Str. 117, D-63500 Seligenstadt, Germany
[2] University of Applied Sciences Emden/Leer, Constantiaplatz 4, D-26723 Emden, Germany
[3] Polytechnic Institute of Bragança, Quinta Sta Apolónia, Apartado 1134, 5301-857 Bragança, Portugal
[4] Artificial Intelligence and Computer Science Laboratory, R. Campo Alegre 102, 4169-007 Porto, Portugal
[5] SAP Research, Vincenz-Priessnitz-Strasse 1, D-76131, Karlsruhe, Germany
E-mails: {marco.joao, armando.colombo}@schneider-electric.com, pleitao@ipb.pt, stamatis.karnouskos@sap.com

Abstract— **The fusion of mechatronics, communication, control and information technologies has allowed the introduction of new automation paradigms into the production environment. The virtualization of the production environment facilitated by the application of the service-oriented architecture paradigm is one of major outcomes of that fusion. On one side, service-oriented automation works based on exposition, subscription and use of automation functions represented by e.g. web services. On the other side, the evolution of traditional industrial systems, particularly in the production area, as a response to architectural and behavioural (functional) viewpoints of the ISA95 enterprise architecture, where a close inter-relation between SCADA, DCS and MES systems facilitate the management and control of the production environment. Automation functions are increasingly performed by the composition and orchestration of services. Among other methods, the application of formal Petri net based orchestration approaches is being industrially established. This paper presents the major characteristics that such a Petri net based orchestration presents when it is developed, implemented and deployed in an industrial environment.**

I. INTRODUCTION

Manufacturing Execution Systems (MES), following the definitions from the MESA (www.mesa.org), offer a set of functions dedicated to monitoring and control production processes on the level 3 of the ISA95 enterprise architecture [1]. Dynamic re-scheduling, production and maintenance planning, production quality management, among other functions, will be implemented having structural and functional interfaces to the level 2, i.e. Supervisory Control and Data Acquisition (SCADA), and level 4, i.e. Enterprise Resource Planning (ERP). It is at these two levels, i.e. SCADA and MES, where decision making processes, supporting the process control and automation, have to be done and where the contributions of this paper are focused.

During the last couple of years, MES and SCADA systems have evolved from centralized components inside the ISA95 to distributed and networked structures following the technological evolution of computer architectures. In fact, they started in the 60´s and 70´s with monolithic centralized components, evolved during the 80´s to distributed structures and in the 90´s and later to networked topologies. Nowadays, such SCADA and MES functions start being provided as services partially located in service clouds [2]. In the case of MES and SCADA systems implemented in a service-oriented manner, functions appear as results of direct exposition of services or of orchestration and composition of services. In all those cases, the orchestration approach, independent of the tools and implementation technology, has to be supported by methods covering runtime decision-making process.

One prominent methodology to specify and implement service orchestration topologies is the use of formal tools following the Petri net (PN) theory tailored for control and automation purposes [3]. In this work, service functions, such as service-exposition, service-call and service-composition, are associated to marking of places, enabling and firing conditions for transitions. This means, basic and fundamental structural and behavioural properties associated to Petri net models, such as deadlock-freeness, sequential and parallel mutual exclusion relationships, cyclic and repetitive behaviours, liveness and marking boundedness (see [4-7] and the references therein) are then mapped into basic and fundamental properties of the modelled orchestration topologies. As a matter of fact, the analysis of the PN-models allows verifying model properties that are then used to validate orchestration topologies.

The objective of this paper is to summarize the major characteristics of a Petri net based methodology to specify, develop and implement SCADA and MES functionalities as a result of a service orchestration and composition in a service-oriented based automation environment.

The rest of the paper is organized as follows: Section II overviews the principles of service orchestration by using the Petri nets formalism, and Section III introduces the proposed approach for service-oriented SCADA and MES systems. Section IV presents the mechanisms for the data acquisition and Section V discusses the mechanisms for event processing. Section VI illustrates the application of the proposed approach for an industrial case study. Finally, Section VII concludes this work and points out possible future related works.

978-1-4673-2420-5/12/$31.00 ©2012 IEEE

II. PETRI NET BASED SERVICE ORCHESTRATION

Since services are not anymore isolated entities exposed by the intervenient software components, new logic control responsible for the interaction among them can be realized using the generic services as building blocks. The model-based orchestration engine should be able to interpret a given work-plan made of services (an orchestration) and execute it. The work-plan can be defined in Business Process Execution Language (BPEL) [9], Petri nets formalism (see for example [3], [10] and [11]), or even in adapted IEC 61131-3 languages, beside others [12]. Another feature is to compose services, i.e. the work-plan itself represents an exposed service.

The modelling language used in this work derives from Petri net specifications, as described in [13]. The developed Petri net orchestration engine has several features, including:

- Lightweight alternative to BPEL and similar to what automation engineers are used to (low learning/adjusting curve).
- Service invocation, exposition and eventing.
- Design time and run-time composition.
- Analysis possibilities of models at design time.
- Integrated decision support for conflict situations on the Petri nets models.
- Interpretation of XML-based configurations (used in dynamic deployment).

Following a Petri net, the orchestration engine needs to know how and when to respond to services and to represent them in the model. This is done by describing transitions in the Petri net model. A transition willing of sending a request/response or an event must be enabled, and the action is done when it fires. In the other hand, a transition receiving a message from a request, response or event, will only fire if it is enabled and the message is there. Fig 1. represents these two types of associations.

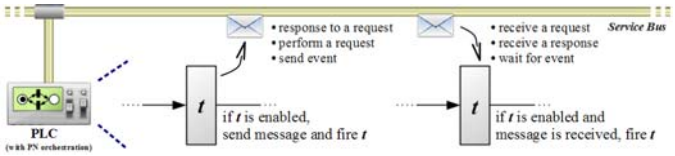

Fig 1. Two types of service association to transitions: transition outputs a message/event (left), transition waits for a message/event (right).

Decision points, alternative paths in a control logic topology, and other kind of sequential or parallel mutual exclusion relationships [5] that appear in the system to be orchestrated, are explicitly modelled by transitions in conflict by the Petri nets, the same way that web services related functions are able to figure out as transitions and places in Petri nets. Such features require the intervention of decision mechanisms. This is especially valid when considering that automation systems are inherited dynamically and not fully predictable; sometimes there are unexpected circumstances that a simple executed Petri net cannot handle: operation's delay and cancelling, synchronization among individual workflows, unexpected situations, unaccomplished operations, dynamically adding new operations, etc.

The following topics describe where additional help is needed in terms of decision, to increase the power of Petri net-based orchestration in automated systems:

- Selection of the firing transition between several ones that are in conflict (resolution of conflicts).
- Petri nets analysis to support decision, including behavioural and structural analysis, path finding and simulation.
- Selection of the best service or operation to execute when the associated transition is enabled or firing.
- Management of the Petri net to decide when to run, stop and reset the system.
- Automatic composition and aggregation of Petri nets.

Fig 2. illustrates an example, described in [13], of using a Petri net model to describe the relation between three machines that are activated when the corresponding transition fires. The machines have web services and when activated, a message is sent to the corresponding machine. The logic here is that only one machine can operate for one request at time. The decision point is translated in the Petri net model as a conflict, but requires that some component of the service-oriented based architecture, basically located in the service bus, resolves the conflict, i.e. choose one of the machines depending on various criteria.

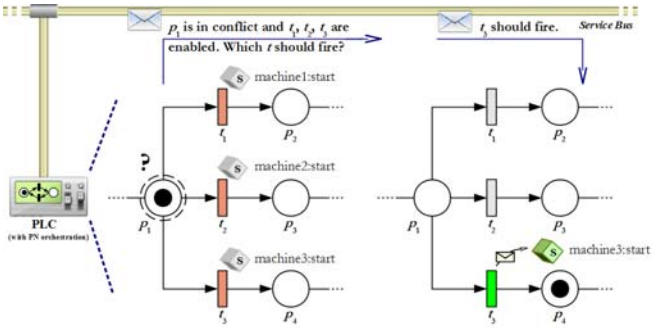

Fig 2. Example of a Petri net orchestration with conflicts.

More information can be consulted in [13].

III. INTEGRATION OF PETRI NET BASED ORCHESTRATION ENGINES IN SERVICE-ORIENTED SCADA AND MES

Service-oriented (event based) paradigm seems to fit very well with the next evolutionary step to empower the typical SCADA systems [2], not only from the organizational point of view in terms of services, but also concerning the flux and processing of events. The preceding implementations of SCADA and MES systems based on traditional service-oriented systems (see for example [3]) demonstrated a service landscape for devices and PC engineering tools with some degree of flexibility for decisions and "plug & play" capability, but events were only used to update the status of devices and for monitoring in PC applications. With the new service-oriented approach, it is intended to fully assess the event-driven approach, i.e. use events as the basis to formulate

actions, e.g., service exposition and service subscription. Therefore, events play a very active role in the architecture and their real-time acquisition and processing is fundamental towards the increased dynamicity of the automation system, which work now in an asynchronous mode.

Translating this to a typical SCADA/MES context, events are generated by the different components in the system and used by the SCADA for the purpose of monitoring, supervision, diagnosis and recovery. These SCADA features will require services that are part of the system to complete their function and possibly generate also new composite services provided by the MES and sometimes by SCADA. This new SCADA differentiates from the typical one as using an event-driven service-oriented manner to supervise processes and resources.

Fig. 3 shows a typical service-oriented SCADA/MES solution. The communication between different resources in the architecture is only possible if common standards are adopted that makes possible not only the share of information (communication) but also the understanding of this information (semantics).

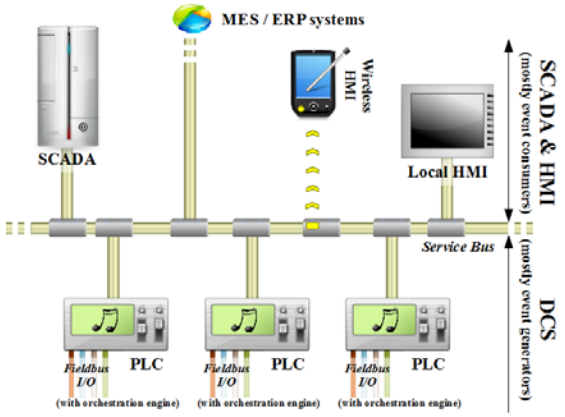

Fig 3. System architecture for service-oriented SCADA and MES.

The service bus, illustrated in Fig. 3, a kind of publisher/subscriber event infrastructure (e.g., Device Profile for Web Services (DPWS) [14]), represents the transparent layer 7 from the OSI model where the means of communication is done in a service-oriented and event driven way. It means that: 1) resources are shared in the network in a form of services that can be requested and orchestrated/composed, and 2) events produced by services that circulate in the network are used to promote analysis and reactions to the current behaviour of the system.

PLCs (Programmable Logic Controllers) and RTUs (Remote Terminal Units) provide the information they acquire as a service for others to consume. They include interfaces for different fieldbus standards and I/O and map it to web services. For example, a RTU interfacing a remote pump station provides a service to control the pumping parameters such as the frequency and pressure. It also generates events to the service bus to indicate what actions are currently taken by the pumping station and to alert of out of desired limit values.

Notably PLCs include the possibility to process logic to coordinate not only sensors and actuators directly connected to the PLC, but also external services, such as the one given in the previous example of the pump station. In the typical manner and using typical PLCs this would be programmed and interpreted by the PLC using the IEC 61131 standard. But the introduction of services also opens the horizon to use new approaches, with distributed service-based control and automation arrangement being referred as orchestration. Therefore orchestration languages can be used and are executed by orchestration engines that are hosted by smart controllers to coordinate internal services and external ones provides by other RTUs and e.g., PLCs.

In the case of having orchestration engines that run deployed Petri nets, there is an exchange of information between two or more smart controllers (e.g., a PLC or a smart I/O) hosting the orchestration engines, performing distributed control/orchestration, SCADA and MES functions (Fig. 4).

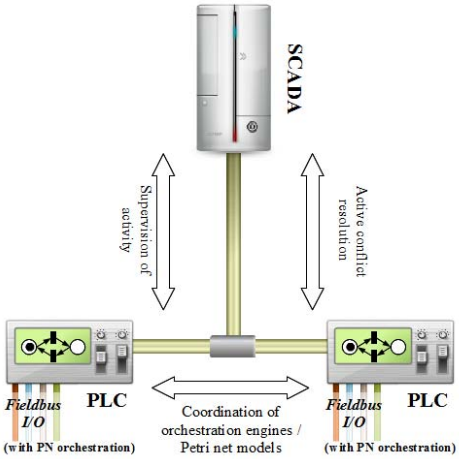

Fig 4. Interaction between PLCs with Petri net orchestration engines and SCADA

Basically, distributed Petri nets running in different orchestration engines require to synchronize their operation using a mechanism like that described in [13]. Having the orchestration engines working, each time that conflicts between enabled transitions appear, representing e.g. decision points in the orchestration logic, events are triggered and forwarded to the SCADA, to inform the current state of activity and to request support for solving those conflicts [8].

The SCADA will actively collect this information and if necessary interact with the MES in order to react to it, supporting a decision mechanism to solve the conflict, e.g., dynamic re-scheduling.

The supervisory and decision system is the kernel of the SCADA / MES system and responsible to acquire events from the service bus. A supervisory system includes an event processing unit running a mechanism to classify and process events that are the basis to formulate diagnostics. Actions are selected and executed by the decision support tools. The automatic processing and monitoring is supported indirectly

by interaction with other systems, and also directly by the system engineers through human-machine interface (HMI).

The main functions of the proposed service-oriented SCADA system are:

- To interact and potentially cooperate with DCS by using the information of events that it makes available.
- To support the DCS by generating new services and pro-actively requesting services to configure, optimize and recover the system.
- To provide information about the system to the users and enable their interaction (via HMI). System configuration can be done via HMI as well as decision-making, if the automatic decision-making processes of the supervisory system are not enough.
- To facilitate the integration and collaboration with other systems, such as MES and potentially even ERP.

IV. Data Acquisition

The acquisition of data in a service-oriented environment means to be aware of the messages in the service bus that circulate between endpoints. This can be done by explicitly capturing them or interacting with the responsible party, e.g., a historical service. The information parsed from messages can then be stored locally into a local database to be used by further processing and also for monitoring (see Fig. 5).

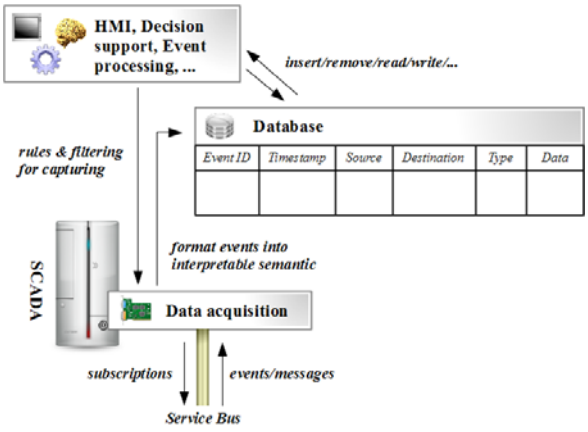

Fig 5. Service-oriented data acquisition from the service bus.

The messages in the service bus indicate events that are happening in the system; also important is to capture messages that are not only with destination to the supervisory system. For this purpose, the promiscuous mode for the associated network interface should be enabled to permit the capture of all messages (even the ones with different destinations).

Concerning the technology side of the data acquisition, i.e. which web service protocols should be used, there are several choices for capturing messages in the service bus. To provide a larger range of compatibility, the data acquisition mechanism doesn't need to be fixed to some profile, but would benefit in being fixed to the basic communication standards and open to profiles and protocols based on those.

If assumed that the data acquisition will capture messages based on e.g. SOAP (Simple Object Access Protocol) (one of the fundamental web services protocol for exchanging information), then it is possible to get events of all profiles that are using it. DPWS [14] and OPC-UA (web services version) [15] are some good examples of profiles that include SOAP for message exchanging. This approach is also compatible with new emerging more lightweight approaches, e.g. REST. In addition, the binary versions of both can also be implemented when a binary capturing mechanism is included in the data acquisition (here it is needed the correct logic for understanding and mapping the binary information).

In the case of DPWS, besides the typical request and response mechanism, it includes a specification concerning eventing, namely WS-Eventing [16]. It can be used to make subscriptions by the supervisory system to different services in the system and get events e.g. about data values whenever these are produced. Filtering of events is also possible, since the Petri net formalism applied in this work considers the association of events to the enabling-firing conditions of transitions. Moreover, dynamic discovery is used whenever devices are connected to the service bus and announce themselves with "*Hello*" events and "*Bye*" events in case they are removed. However, infrastructure support can also be present for dynamic discovery and messaging exchange support.

In any case, there are a lot of possibilities to capture event messages from the service bus and their information can be parsed in a very basic level to identify some useful parameters. Depending on what profiles and protocols are used for implementing the service-oriented approach, different parameters are desirable, such us unique event identifier, timestamp referring to the instance when it was generated and when it was captured, and source and destination of the event.

In the case of a Petri net orchestration engine, events are generated under the following circumstances:

- Transition is enabled.
- Transition has fired.
- Passive conflict: does not request a resolution but does accept orders from the supervisory system.
- Active conflict: requests a resolution from the supervisory system.
- Executed actions and services associated to transitions;
- Start/ Stop of the orchestration engine.

The captured information will be afterwards used by the event processing and decision support mechanism. The stored events can be manipulated in the database after processing and used to take actions. Moreover, a HMI connected to the supervisory system is possible to monitor the events registered in the database that mirrors what is happening in the lower-level of the control architecture.

V. Event Processing

The event processing is used to take the registered events, assess them and derive possible actions; additionally, it may compose events that can be used for example, as a basis for the decision support system (see Fig. 6).

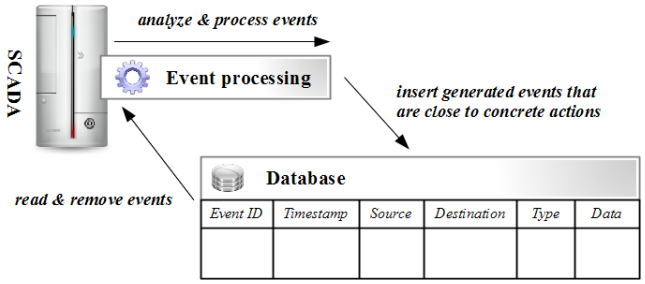

Fig 6. Event processing and interaction with the event database.

These advanced events can be a composition of simpler events captured by the data acquisition or even completely new generated events based on system analysis and diagnostics. For example, based on the received events, it was detected that the production capability of a product is suffering further delays. Analysing the acquired "raw" events, the event processing tool concludes that there is a bottleneck in the DCS concerning the orchestration of different services provided by RTUs. In consequence, the event processor generates a new event to indicate that there is a situation to be improved.

For event processing, different methodologies are possible:

- Data mining: depending on how many events exist and on the complexity of the system, data mining would provide a first identification and classification of events, e.g., grouping similar events into clusters.
- Composition: composing event information can be done to "summarize" a set into one more meaningful (in a larger context) event. For example, "heartbeat" events that are periodically sent by on device can be composed into one indicating that this device is online.
- Conflict resolution: some situations during the orchestration of services will require that one of several paths must be selected to move on with the orchestration. This event is reported and can be further processed to be handled to the decision support system to make a decision upon.
- Optimization analysis: events also report information about the status of different parameters. In this case, the trend of a parameter can be evaluated and compared with optimal or expected values. If the tendency is below the threshold, optimization request events can be generated to report an improvement for the parameter (essential in case of energy management where energy consumption values need to be optimized).
- Diagnostics: if something is reported to have an undesired behaviour in the DCS, the reason and consequent action can be sometimes found in reported events by the system. The event processor, when receiving a diagnostics event, needs to make a tracing in the event database to find the source of the problem. From the diagnostics, events can be generated with the information of the diagnostics to be further processed and actions taken based on the diagnostics data.
- Complex Event Processing (CEP): processing of large sets of events that happen in an organization and taken consequent actions in real-time may require sophisticated analysis and assessment both of which can be handled by CEP.

As with data acquisition, event processing can be as well monitored and configured by local or remote HMIs that provides the tools for user interaction and therefore enable the interaction with human operators in the process. Assisted or unassisted event processor creates a set of conditions for making decisions in the DCS that will be handled by the decision support system of the SCADA.

VI. Industrial Use Case

The scenario depicted in Fig. 7, and previously used in [3], is used to demonstrate how the SCADA/MES features for smart controllers with embedded Petri net orchestration engines can be applied in an industrial environment. It represents a flexible assembly system composed of a transport system made of mechanical conveyor modules (C1-C11), lifters (L1, L2) and workstations (W1, W2). The arrows identify the transfer capability of a conveyor/lifter with its neighbours. Pallets enter in the system via L1 and can leave it via L2. In C2 and C8 pallets can be stopped to be operated by a machine or similar workstation (W1 and W2).

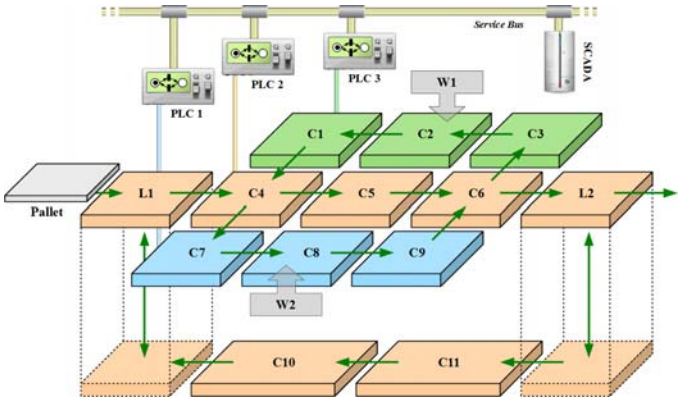

Fig 7. Service-oriented infra-structure of the experimental cell.

For the identification of pallets, the cross units C4, C6 and the workstation units C2, C8 are equipped with RFID (Radio-Frequency IDentification) readers that are able to read/write information from/to tags attached to the pallet. The identifier will be used by the orchestration engine to "ask" an external system for the next production step for a product. A next generation would not ask at each crossing section, but would store the production history directly on the tag.

A. Application of the Service-oriented Approach to the Use Case

The service-oriented system behaviour is represented in the orchestration topology, modelled by using the Petri nets formalism and designed according to the available atomic services (and their operations) and the pretended behaviour of the devices, representing their control logic. Note that atomic services are exposed by the transfer units, lifters and RFID devices. In this work, 3 orchestration engines were developed:

one running in controller #3 for the components C1-C3, other running in controller #2 for the components C4-C5, L1, L2, C10-C11, and another running in controller #1 for the components C7-C9. The generated orchestration models communicate with each other (for inter transfer operation of pallets) using the service invocation (i.e. the "*TransferIn/TransferOut*" mechanism).

Giving that the orchestration topology has been formally specified by a Petri net model and that this model is being running in the orchestrator/orchestration engine embedded into a smart device or system within a service-oriented automation architecture, the model contents a considerable amount of information, which can be useful applied to support SCADA and MES functions. Model-based monitoring indexes are supplied by e.g., transition flows and place-flows. This means, the complete set of results of a structural analysis of the PN-models are inherent monitoring data of the orchestrated process and implicitly rich contents of the service-oriented SCADA.

As a matter of fact, the transition flows are the basic orchestration/composition topologies offered by the service-oriented automation processes. Remark: as usually in linear algebra and functional analysis, having the basis of the vectorial space, the whole space can be generated by vector composition. The decision-making system, as part of the service-oriented MES, exposing scheduling functionalities, can at any time select some of those transition-flows, a sequence of them and also different composition of them, to decide the most recommendable orchestration topology for a given state of the service-oriented automation process.

One of the major outcomes of this approach is the capability of the service-oriented automation system to use results of a formal analysis and validation of the Petri net model-based orchestration for enhancing the SCADA, MES and also DCS functions. That is, the service-oriented automation process is orchestrated by a Petri net model, which is not only exposing and consuming control services, but also supporting SCADA and MES functionalities like monitoring services, scheduling and dispatching services and error-recovery services.

Another major outcome of the presented methodology is that the orchestration modules deployed into smart automation devices allow providing supervisory control services, such as monitoring or other SCADA functions. These new services are result of the orchestration running in those devices. Note that as result of orchestration process, new services appear, representing behaviours that were not exposed by the existing components, i.e., emergent behaviours.

When the orchestration process is running and the models are running the token-game, conflict situations formally specified by set of transitions in conflict (in the Petri net models) are reported and have to be solved in runtime conditions. Fig. 8 illustrates the modelling of decision on the workstations associated to conveyors C2 and C8, appearing when the place "loaded" is marked.

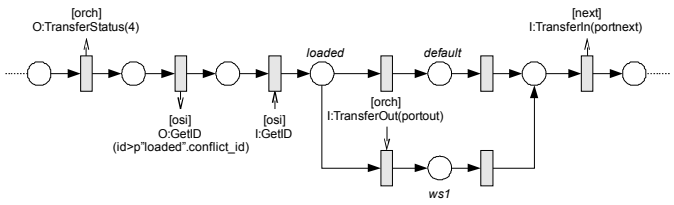

Fig 8. Modeling decisions on workstations.

One important aspect of this approach is that the SCADA or MES systems can be considered as central components like the structures illustrated in Fig.3 and Fig. 4. Nevertheless, the smart service-oriented compliant automation devices are having SCADA and MES functionalities embedded inside. As a major result, distributed SCADA/MES functions are also exposed as services by devices and systems in a distributed fashion. This implies that the conflicts can be remotely handled by a centralized service-oriented based SCADA/MES or by locally embedded SCADA/MES systems.

When a new SCADA/MES function is needed to solve a conflict, and it is not exposed as service inside the existing service-oriented compliant smart devices and systems, this new function can dynamically be uploaded inside of some existing smart devices. In this case, the service-oriented SCADA/MES is presenting a very important characteristic, i.e. evolvability. At any time, SCADA/MES functionalities exposed as services and also orchestration services that are not necessary can be deleted or brought into stand-by state (not accessible/exposable), without changing the physical architecture of the system.

B. Engineering Methodology and Tool-Chain to perform Petri Net-based Orchestration

A SoA-compliant device, which is having WS-embedded capabilities and is dedicated to compose or orchestrate services exposed by itself and/or by other devices of the SoA-based automation architecture, needs to have an Orchestration Engine. In the approach presented in this paper, such an orchestration engine consists, e.g. of a Petri Net Interpreter, which is configured in such a way that it performs the token-game of the PN and the evolution of the net corresponds to the composition and orchestration topologies.

A complete methodology and engineering tool-chain called "Continuum" (see [3]) has been developed, implemented and deployed into an industrial prototype application. Fig. 9 depicts the major characteristics of such an engineering system. As one of the major outcomes of this development it is possible to identify the existence of a "Petri Net Orchestration Tool and Engine" which also can be used as "Source" of a Service (Orchestration Service) to be exposed and also consumed by devices and systems inside the SoA-automation architecture.

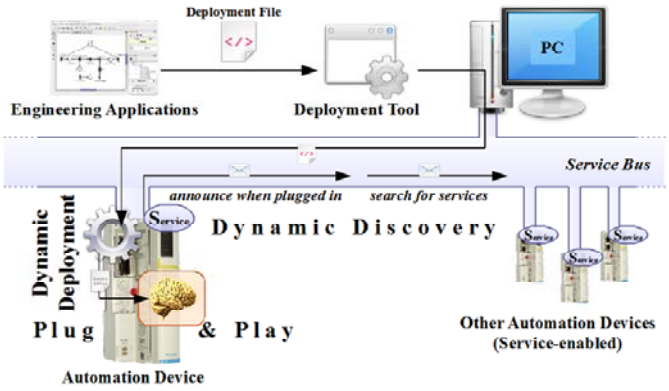

Fig 9. Engineering Methodology and Tool-Chain for a PN-based Orchestration.

VII. Conclusions and Future Work

The proposed approach considers Petri Nets based orchestration engines embedded in industrial controllers that orchestrate the services offered by the physical devices to support SCADA and MES systems.

An important aspect enabled by this approach is the multi-layer collaboration based on service-oriented principles while individual decision making processes (e.g. via Petri Nets) are hidden from the overall interaction. Nevertheless the intriguing part is that the respective decision making processes can now consider the collaboration/interaction with other layers (e.g. MES). Moreover, it can integrate knowledge or restrictions from those other layers to their functionalities, in an event-driven approach. For this to be possible, of course both a communication infrastructure as well as semantic understanding of the exchanged information in a specific context is a must. As the next generation of SCADA/MES systems considers that portions of their functionality may reside in a cloud [2], such interactions are expected to be more common (no communication barriers) but also more challenging (in finding the right level of performance vs. functionality vs. security etc.).

Acknowledgment

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement 258682 (IMC-AESOP: *ArchitecturE for Service-Oriented Process - Monitoring and Control*) and 224053 (CONET: Cooperating Objects NETwork of excellence).

References

[1] http://www.isa-95.com

[2] S. Karnouskos, and A. W. Colombo, "Architecting the next generation of service-based SCADA/DCS system of systems", 37th Annual Conference of the IEEE Industrial Electronics Society (IECON 2011), Melbourne, Australia, 7-10 Nov 2011.

[3] J.M. Mendes, A. Bepperling, J. Pinto, P. Leitão, F. Restivo, A.W. Colombo, "Software Methodologies for the Engineering of Service-oriented Industrial Automation: The Continuum Project", Proceedings of the 33rd Annual IEEE International Conference on Computer Software and Applications (COMPSAC'09), pp. 452-459, 2009.

[4] T. Murata, "Petri Nets: Properties, Analysis and Applications", *IEEE*, vol. 77, n. 4, pp. 541-580, 1989.

[5] M. Zhou, F. DiCesare, "Petri net synthesis for discrete event control of manufacturing systems", The International Series In Engineering and Computer Science, Kluwer Academic Pub. (Boston), 1993.

[6] R. David, H. Alla, "Discrete, continuous and hybrid Petri nets", Springer Verlag, 2010.

[7] A.W. Colombo, R. Carelli, B. Kuchen, "A temporized Petri net approach for designing, modelling and analysis of flexible production systems", International Journal of Advanced Manufacturing Technology, vol. 13, n. 3, Springer Verlag London, pp. 214-226, 1997.

[8] K. Feldmann, A.W. Colombo, "Material flow and control sequence specification of flexible production systems using coloured Petri nets", *International Journal of Advanced Manufacturing Technology*, vol. 14, n. 10, Springer verlag London, pp. 760-774, 1998.

[9] NN, "Web Services Business Process Execution Language Version 2.0", OASIS Standard, April 2007.

[10] R. Hamadi, B. Benatallah, "A Petri net-based model for web service composition", Proceedings of the 14th Australasian database conference, Darlinghurst, Australia, pp. 191-200, 2003.

[11] L. Bing, C. Huaping, "Web service composition and analysis: a Petri-net based approach", 1st International Conference on Semantics, Knowledge and Grid (SKG '05), November 2005.

[12] K. Feldmann, A.W. Colombo, C. Schnur, T. Stoeckel, "Specification, design and implementation of logic controllers based on coloured Petri net models and the standard IEC1131. Part I: specification and design", IEEE Trans. on Control Systems Technology, pp. 657-665, 1999.

[13] J. M. Mendes, P. Leitão, A.W. Colombo, F. Restivo, "High-Level Petri Nets control modules for service-oriented devices: A case study", Proceedings of the 34th Annual Conference of IEEE Industrial Electronics (IECON'08), pp. 1487-1492, 2008.

[14] http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01.

[15] W. Mahnke, S.-H. Leitner, M. Damm, "OPC Unified Architecture", Springer, 2009.

[16] D. Box, et al, Web Services Eventing (WS-Eventing), http://www.w3.org/Submission/2006/SUBM-150 WS-Eventing-20060315/, 15 March 2006.