View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Biblioteca Digital do IPB

Process Optimization of Service-Oriented Automation Devices Based on Petri Nets

J. Marco Mendes[1], Paulo Leitão[2,4], Francisco Restivo[1,4], Armando W. Colombo[3]

[1]University of Porto - Faculty of Engineering, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

[2]Polytechnic Institute of Bragança, Quinta S^{ta} Apolónia, Apartado 134, 5301-857 Bragança, Portugal

[3]Schneider Electric Automation GmbH, Steinheimer Str. 117, D-63500 Seligenstadt, Germany

[4]LIACC – Artificial Intelligence and Computer Science Laboratory, University of Porto, Portugal

E-mails: {marco.mendes,fjr}@fe.up.pt, pleitao@ipb.pt, armando.colombo@de.schneider-electric.com

***Abstract* – This paper introduces a novel method for the specification and selection of criteria-weighted operation modes for the orchestration of services in industrial automation using Petri nets. The objective is to provide to the internal decision support system of a service-oriented automation device or of another applicable computational system the capability to select the best path in a Petri net orchestration model considering different criteria to evaluate the quality of services, such as the time, energy efficiency and reliability. The transition-invariants obtained from the Petri net represent the set of possible modi operandi and these are then weighted with decision criteria. The result will be afterwards evaluated in order to select the optimal modus operandi to be executed by the device. Based on the experiments, this method permits the dynamic optimization of processes in real-time, considering available parameters from devices and other resources.**

I. INTRODUCTION

Service-orientation principles are pointed out as a promising solution to address the current challenges in industrial automation and production systems design and operation, namely the modularity, flexibility and re-configurability. Standardized services and the advanced separation of interfaces and implementation, enhance the abstraction of component-based development and thereby pave the way for non-technical software engineers to develop complex, process-oriented software systems [1].

In order to reach a level of availability and integration of this technology, services must also be available in industrial controllers. Some of the first steps were done by the definition of Devices Profile for Web Services (DPWS) [2] and its implementation into industrial devices. DPWS defines a profile over a specific set of web services protocols to enable secure web service capabilities on resource-constraint devices [3]. Therefore, simple or complex services can be called directly by other devices or enterprise information systems [4].

However, it is not expected that such devices be only able to provide services representing their resources, but also a source of multi-functional actions concerning service-orientation. Particularly, composition and orchestration have been seen as the form of engineering of service-oriented architectures, and the inclusion of these features in industrial devices is still a major effort. The representation of the work-plan associated to services, to be interpreted and executed by orchestration engines, can be defined using different methods [5], namely the Business Process Execution Language (BPEL) [6] and the Petri nets formalism (see the work of Hamadi and Benatallah [7] and Deng et al. [8]). In this work, the selected modeling language is Petri nets taking advantage of its powerful mathematical foundation that will support, among others, the analysis and validation during the design phase and the decision-making for conflict resolution.

Most research works have been concerned with the co-ordination of services, specially the automatic way of creating new orchestrations based on available services and some rules on how to compose them and to generate new forms of services. There are several methodologies for that purpose, since the use of semantic services [9-11] to the application of intelligent systems (such as multi-agent systems [12]) to support the construction of workflows from services (e.g. using BPEL [13]). Evaluation of services and the use of quality of service (QoS) is also used when generating orchestrations and selecting the best possible service [14-15].

Once workflows are available to be executed and since they describe mostly all possible combinations of available processes (modi operandi), there are still decisions required in selecting the best process (modus operandi) in a specific circumstance. For instance, a pallet has the option to be conveyed straight ahead or to the right (requesting the corresponding service from the transport system). The answer can be given based on required manufacturing services, energy consumption, speed, and other quality parameters. Consequently, the decision of the best modus operandi is a key issue to improve the system performance that depends always on current situation of the automation system.

This paper addresses this issue by introducing a novel approach to the real-time decision making in service-oriented systems, considering the structural knowledge extracted from the Petri nets models (in this case transition-invariants), combined with a flexible set of decision criteria. This permits that at runtime the device or another computational device used in industrial automation is able to analyze a defined workflow of services and select the best possible modus operandi based on the specified decision criteria.

The reminder of this paper is organized as follows: first, section 2 overviews the basic concepts of service-orientation related to automation devices and Petri net-based orchestration engines. Section 3 introduces the proposed procedure for the process optimization based on the Petri nets knowledge combined with a flexible set of decision criteria, and section 4 illustrates the application of the proposed method into an experimental case study. Finally, section 5 rounds up the paper with conclusions.

II. Service-Oriented Automation Devices with Petri Net-based Orchestration Engine

Service-oriented architecture (SOA) was seen as a new ground for experimentations in industrial automations since it relative success in the business chapter from the beginnings of the 21st century. The emergence of SOA in automation domain and the use *web services* standards became notorious after the successful application in automation devices and as a new form of engineering (see SIRENA [16] and SOCRADES [17] projects). However, a major industrial acceptance, besides the research projects scope, is needed, due to the lack of demonstrated features of both automation devices and supporting applications.

The main difference to the other technologies does not only rely on the implementation of the basic resources (in SOA these resources are called atomic services), but in the way that they are used and composed into complex applications. A main requirement is the richness description of a service, so that it can be correctly used afterwards by a client. Therefore, *Web Services Description Language* (WSDL) [18] is the main protocol that is used to define the interface of the service by its elements (e.g. operations, types, inputs, outputs). On top of the description, model-based *orchestration* defines a work-plan made of services to be executed.

The modeling language used along this paper to describe service processes derives from Petri net specifications (see [19] for more information). The developed Petri net orchestration engine has several features, including:

- Lightweight alternative to BPEL and similar to what automation engineers are used to;
- Service invocation and exposition;
- Design time and run-time composition of orchestration models;
- Analysis possibilities of models at design time;
- Conflict resolution at run-time and integrated decision support for conflict situations on the Petri net models
- Interpretation of XML-based configurations (used in dynamic deployment).

A major task at this stage is to fit the orchestration engine and web service technology into an automation device. The resulting *smart embedded devices* (demonstrated in the SOCRADES project) are the host for the most of the services exposed in the system and also responsible for the coordination and control activities (see Fig. 1). They include an *orchestration engine* to "link" services together and to create new composite services. Atomic services representing resources and functions of the connected equipment are provided by the *device interface*. An internal *decision support system* is responsible to sustain the engine for decisions, e.g. selecting the best modus operandi based on decision criteria.

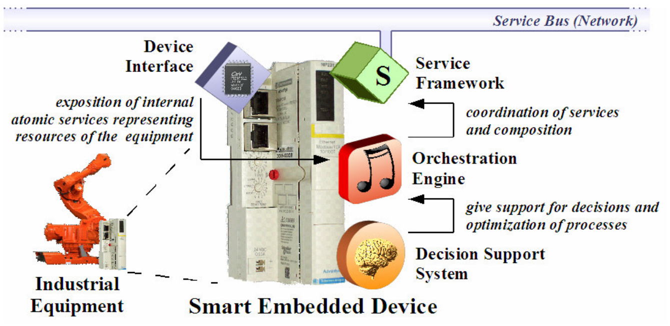
Service Bus (Network)
Device Interface
Service Framework
exposition of internal atomic services representing resources of the equipment
coordination of services and composition
Orchestration Engine
give support for decisions and optimization of processes
Decision Support System
Industrial Equipment
Smart Embedded Device

Fig. 1. Structure of a smart embedded device.

Computer tools are necessary to configure devices. This includes modeling and planning software, analysis utensils, device and service deployment tools and also posterior monitoring applications. The Continuum Development Tools (CDT) [20] was developed with the aim to facilitate these activities. The main component is build around the Continuum Development Studio (CDS) that is based on an extensible Document/View framework, provides an engineering tool for service-oriented automation entities, for example, supporting the visual description, analysis, simulation and deployment of their behavior in Petri nets formalism.

III. Process Optimization of Service Operations

The general approach of the proposed methodology for the process optimization of service operations is represented in Fig. 2. It is based on the necessary steps on the design phase (when workflows are defined and configured) and operation phase (when workflows are executed at runtime by devices). Note that these two phases only include the procedures that are important to this decision and optimization method (since the design and operation phase involving devices requires more steps than presented on this work).

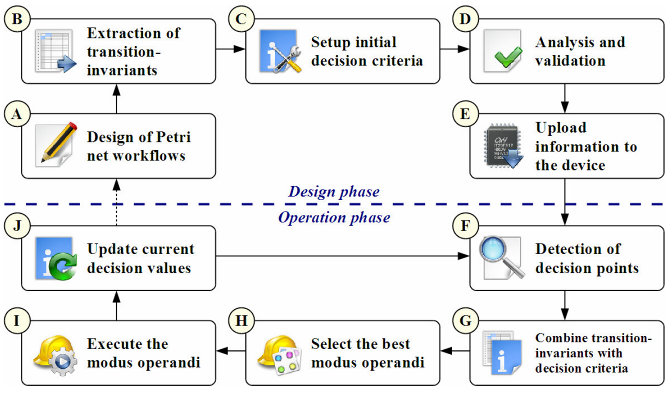
B
Extraction of transition-invariants
C
Setup initial decision criteria
D
Analysis and validation
A
Design of Petri net workflows
E
Upload information to the device
Design phase
Operation phase
J
Update current decision values
F
Detection of decision points
I
Execute the modus operandi
H
Select the best modus operandi
G
Combine transition-invariants with decision criteria

Fig. 2. Procedure for the process optimization based on Petri net workflows and decision criteria.

The several steps of the procedure will be explained in the next subsections.

A. *Design of Petri net workflows*

The processes to be analyzed and executed by automation devices are represented by the Petri net formalism (according to the definition of T. Murata [21]). A Petri net is a 5-tuple, $PN = (P, T, F, W, M_0)$ where $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places, $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation), $W: F \to \{1, 2, \ldots\}$ is a weight function, $M_0: P \to \{0, 1, 2, \ldots\}$ is the initial marking, $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$. The Petri net structure without specifying the initial marking is denoted by $N = (P, T, F, W)$.

Some of the transitions are linked to the request or provisioning of basic device services. A service's operation is then triggered when the corresponding transition enables/fires. In this case, $S: T \to \{s_1, s_2, \ldots, s_n\}$ represents the finite set of services' operations associated to corresponding transition. A specific $s \in S$ can be empty (meaning there is no operation associated to the transition) or a label identifying the service and its operations. A service and a corresponding operation will be expressed as *service.operation*[*in*|*out*](*parameters*). The *in*|*out* reference indicates if the operation is a request or a response (i.e. incoming or outgoing message in the perspective of the server and outgoing or incoming message in the viewpoint of a client). Messages can be added with information, represented by the *parameters* field.

The design of the Petri nets and the association to services can be done with the CDS tool. WSDL files, representing the description of services, can be imported and the contained operations are listed in order to be associated to the transitions of the Petri net. Other extensions and features of Petri nets can be used as well, but they are not discussed here because of being out of scope.

B. *Extraction of transition-invariants*

In order to extract the transition-invariants of a Petri net, its structure N is used. For the structural analysis, it is important to firstly obtain the incidence matrix of the Petri net. For a Petri net N with n transitions and m places, the incidence matrix $A = [a_{ij}]$ is an $n \times m$ matrix of integers and its typically given by $a_{ij} = a_{ij}^+ - a_{ij}^-$ where $a_{ij}^+ = w(i, j)$ is the weight of the arc from transition i to its output place j and $a_{ij}^- = w(j, i)$ is the weight of the arc to the transition i from its input place j. A^T represents the transpose of the matrix A.

An integer solution x of the homogeneous equation $A^T x = 0$ is called a transition-invariant. The analysis of the transition-invariants allows the identification of work cycles in the Petri net model. There are several algorithms to resolve the equation and determine the minimal set of solutions, i.e. transition-invariants (see for example, C. Amer-Yahia et al. [22]).

The extraction of transition-invariants, as well the place-invariants, can be performed in the CDS.

C. *Setup initial decision criteria*

Decision criteria can be defined for each service $s \in S$ using several attributes $A_s = \{a_1, a_2, \ldots, a_k\}$. Since attributes are possibly of different units of measurement, normalization has to be done. In this case, the adopted procedure is to convert each attribute $a \in A$ to a fuzzy interval of [0, 1] where the maximization of this value is considered.

In this case, the linear normalization is given as an example. Other normalization approaches can be used as well such as the exponential and logarithmical. For the normalization of a value v into n ($n \in [0, 1]$), the desired maximum and minimum of the attribute must be known (v_{max} and v_{min}, $v_{max} > v_{min}$). If the quantity is directly proportional to the normalization interval [0, 1], i.e. the quantity is considered better the higher the value is, then linear normalization can be achieved by

$$n = (v - v_{min})/(v_{max} - v_{min}) \, , \, v_{min} \leq v \leq v_{max}$$

From the other hand, in case of inverse proportionality, the normalization must be done using

$$n = 1 - (v - v_{min})/(v_{max} - v_{min}) \, , \, v_{min} \leq v \leq v_{max}$$

Fig. 3 shows and industrial lifter to lift pallets via the two ports (that may be connected to conveyors). The lifter has a transfer service with two operations responsible to transport a pallet from port A to port B and vice-versa. Each one of the operation has defined attributes to be used as decision criteria. In the example, energy efficiency, quickness of operation and reliability are defined. They have different values for each operation that may be gathered from previous experiences, defined initially using vendor specific information, or just defined for example purposes (as in this case).

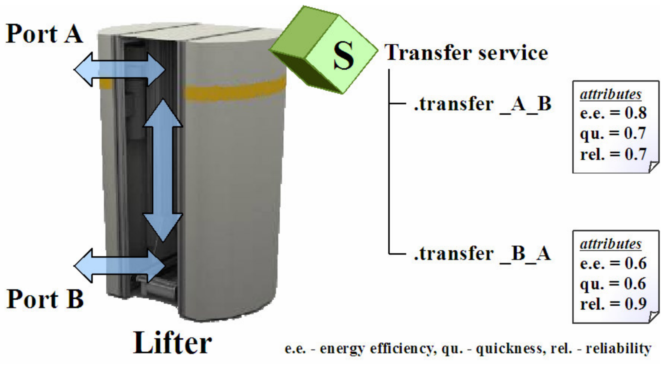

Fig. 3. Industrial lifter with a transfer service and current operation attributes for the decision criteria.

In Fig. 3, the mean quickness of the operation from port A to port B (*.transfer_A_B*) was defined as 11 seconds and from port B to port A (*.transfer_B_A*) is 12 seconds. Considering that maximum and minimum values for this attribute are, respectively, 18 and 8 seconds, and that the quickness is inversely proportional to the normalization quantity (less time means better value), the quickness values for *.transfer_A_B* and *.transfer_B_A* will be 0.7 and 0.6. This means that from the speed point of view, *.transfer_A_B* would be the selected operation because of the higher value (0.7).

Decision criteria should also be changed at run-time to provide an update to the current situation of the system, especially when involving a learning system that can balance the attributes according to the past situations. For example, the energy efficiency of an equipment will probably be reduced with its increasing age.

D. Analysis and validation

The analysis and validation of the models, considering the decision criteria, is used to verify its correctness and the effects of the decision criteria into the final decisions. For this purpose, discrete simulation is performed, after which conclusions can be extracted to support the validation and/or optimization of the model for execution.

Behavioral and structural analysis as well as step-wise simulation can also be performed in CDS tool.

E. Upload information to the device

Since Petri net models are designed and analyzed offline, all the modeling information has to be formatted in a device-interpretable semantic, so that it can be uploaded and interpreted by the device. This step is responsible for configuring the device with the previously defined information (Petri net model, transition-invariants, decision criteria etc.). Once successfully completed, the device can start running, corresponding to the operation phase.

With the CDT it is possible to configure automation devices with the enabled Petri net orchestration engine. An implemented feature from DPWS is implemented that permits the dynamic deployment of services into the devices as well as the general configuration of the device. The only requirement is that the device must be ready, attached to the network and discoverable by the tools.

F. Detection of decision points

Detection of decision points can be done when they actually happen during the execution of the workflow or analyzed previously when the model is about to be executed. In any case, the decision points represent situations where there is a need of the decision support system to provide a concrete answer to the execution system of the workflow.

In terms of Petri nets, decision points are identified by conflicts in the Petri net (see Fig. 4). There is the possibility to model Petri nets without conflicts, but the existence of such properties creates a new dimension in terms of flexibility of Petri nets. Besides static models that only specify a predefined work-plan, some models can be enriched with the possibility of choices that permit the intervention of decision systems.

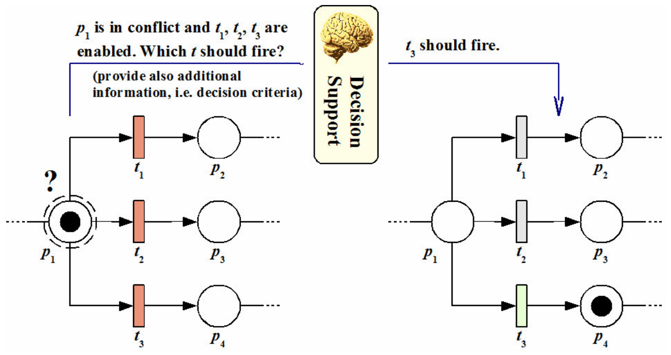
p_1 is in conflict and t_1, t_2, t_3 are enabled. Which t should fire?
(provide also additional information, i.e. decision criteria)
Decision Support
t_3 should fire.
?
t_1
t_2
t_3
p_1
p_2
p_3
p_4
t_1
t_2
t_3
p_1
p_2
p_3
p_4

Fig. 4. Conflict detection/resolution in the Petri net orchestration engine.

A place p has a *structural conflict*, $SC(p)$, if there are at least 2 transitions $t \in T$ where $w(p, t) > 0$. The set of all structural conflicts is denoted by SC. A place p is in *conflict*, $C(p)$ if it has a structural conflict $SC(p)$ and the current number of tokens of the place p, $M(p)$, enables at least two transitions t, which the place p is input.

Structural conflicts are quite suitable since they represent the candidates for the real conflicts that happen at runtime. In fact, the set of real conflicts C is always a subset (or equal) of the structural conflicts SC, $S \subseteq SC$. These candidates can be obtained by the structural information of the Petri nets model and if calculated at designed phase, performance is increased afterwards (avoiding the analysis of each place).

G. Combine transition-invariants with decision criteria

For a given decision point, the decision support system will now combine the pre-calculated transition-invariants of the workflow with the current decision criteria.

For a given transition $t \in T$ associated to a service operation $s \in S$, the combined attribute value for t is given by

$$A(t) = \frac{a_1(t) + a_2(t) + \ldots + a_k(t)}{k}$$

where $a_i(t)$ is the normalized value of an attribute of the service s associated to the transition t. There are k different attributes to be considered for the transition t.

The decision factor of a transition-invariant (modus operandi) x extracted from a Petri net workflow is given by

$$F(x) = b\,\frac{\sum(C_x A_x)}{\sum(C_x)}$$

where C_x represents the set of non-null coefficients of all $t \in x$, A_x is the set of combined attribute values of all non-null coefficient $t \in x$. The value of $b \in [0, 1]$ indicates how much of the decision factor of x is to be considered (0 means not to be considered and 1 fully considered). Similarly, the values of attributes a_i and combined attributes $A(t)$ can also be weighted by a $w = [0, 1]$ before each operation. This represents the weight the attribute's value has in the final decision.

H. Select the best modus operandi

Once the decision factors are calculated for each transition-invariant, the selected modus operandi would be the one corresponding to the transition-invariant with higher decision factor. For example, if $F(x_1)$ and $F(x_2)$ are two decision factors for respectively x_1 and x_2, and $F(x_1) > F(x_2)$, then x_1 will be modus operandi selected.

I. Execute the modus operandi

The selected modus operandi will be executed by triggering the transitions associated to the selected services workflow. The non-selected modi operandi can be minimized (e.g. enter standby modus).

J. Update current decision values

After the decision-making process and posterior execution of a service, new values for the attributes can be determined and balanced with the previous ones.

Decision values, as the whole information of the Petri net is part of the Petri net orchestration engine and therefore are stored locally at the device. Updated values can be obtained directly via the input/output interface of the device (if it is connected to industrial equipment) or requested via the network (as a service operation or subscribed service event).

IV. Experiments

Aiming to illustrate the proposed the concepts, besides the previous formal definition, an example is used as shown in Fig. 5. The case study comprises two conveyors to transport pallets to and from *Machine 1* and *Machine 2* that perform some type of production operations over the objects that are on the pallets. The system is a simplification of a demonstrator used in the SOCRADES project (see [19-20]) just for the purpose of proving the introduced methodology.

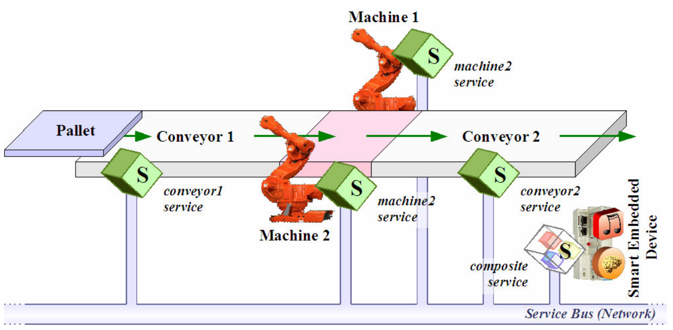

Fig. 5. Production and transport system used as example and application of the methodology (including the representation of the services).

All of the four devices (*Conveyor 1*, *Conveyor 2*, *Machine 1* and *Machine 2*) are connected to the network and expose their capabilities and resources in form of services, as represented in Fig. 5. The Table I shows the characteristics of each equipment and the associated service. Each service has a set of operations; in this case and for simplification purpose, all services have a *.start*[*in*] and *.finished*[*out*] operation (that will request and make the response of the service). Besides that, the equipments also have several criteria attributes to be used by the methodology. A final composite service does the coordination of the system by using the available services from the equipments in a logical way.

TABLE I
Characteristics of the Equipment and their Services

Equipment (Service)	Attributes (normalized)	
	Energy efficiency	Production quality
conveyor1	0.6	-
conveyor2	0.7	-
machine1	0.8	0.5
machine2	0.3	0.7
composite	-	-

Fig. 6 represents a Petri net model which its execution can be requested by the service *composite* (t_1 indicates the start of the service and t_{10} when it is finished). The execution comprises the execution of the *conveyor1*, then the selection of the *machine1* or *machine2* services at the place p_4. Once decision is taken over one of the services, the operation is started via t_4 (t_5) and finished via t_6 (t_7) for the *machine1* (*machine2*). Finally, the operation of *conveyor2* is called.

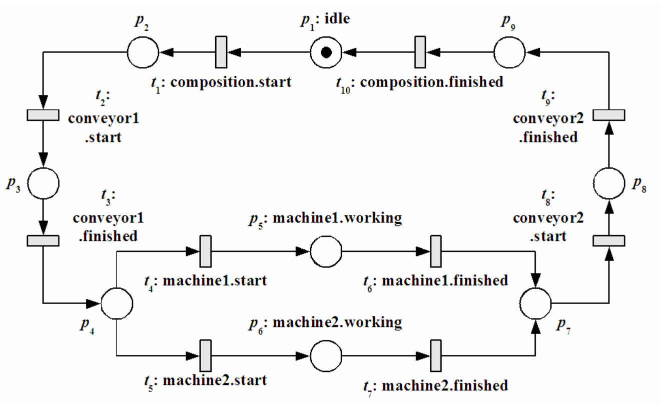

Fig. 6. Petri net model with the composition of the four equipment services (conveyor1, machine1 or machine2, conveyor2).

Using the example of Fig. 6, the transition-invariants of the model are the following:

$$x_1 = t_1 + t_2 + t_3 + t_4 + t_6 + t_8 + t_9 + t_{10}$$

$$x_1 = t_1 + t_2 + t_3 + t_5 + t_7 + t_8 + t_9 + t_{10}$$

These solutions (x_1 and x_2) represent the available modi operandi. In the case of x_1, after *composition.start* is requested, *conveyor1* is called (using *start* and *finished* operations), *machine1.start* is called (waiting then that the *machine1.finishes* its operation), *conveyor2* is coordinated via *start* and *finished* operations and finally *composition.finished* indicates the termination of the modus x_1.

In the scenario Fig. 5, the several service operations associated to the transitions may have decision criteria, such as the attributes of energy efficiency (a_1) and production quality (a_2). For example, transition t_4 represents the start of *machine1* and its operation. Its current energy efficiency value is $a_1(t_4) = 0.8$ (means that energy consumption is minimum) and production quality is $a_2(t_4) = 0.5$ (a reasonable quality). The *machine2* started by transition t_5 may have $a_1(t_5) = 0.3$ (less energy efficient than *machine1*) and $a_2(t_5) = 0.7$ (better production quality as *machine1*). The conveyors only have the attribute for energy efficiency, in this case $a_1(t_2) = 0.6$ and $a_1(t_8) = 0.7$.

Once the system is running, decision points must be detected to be able to fire the right transitions associated to the conflicts. In this case a single decision point is located in the place p_4, where either the path via *machine1* or via *machine2* has to be selected. For this purpose, two attributes (energy efficiency and production quality) are considered for transitions t_2, t_4, t_5 and t_8. The combined attribute values for t_2, t_4, t_5 and t_8 are:

$A(t_2) = [0.6] / 1 = 0.6$ $A(t_5) = [0.3 + 0.7] / 2 = 0.5$

$A(t_4) = [0.8 + 0.5] / 2 = 0.65$ $A(t_8) = [0.7] / 1 = 0.7$

The other transitions do not have attributes and their combined attribute value is 0. Since there are two modus operandi x_1 and x_2, the decision factors are

$$F(x_1) = 1 \times \frac{\sum((1, 1, 1) \times (0.6, 0.65, 0.7))}{\sum(1, 1, 1)} = 0.65$$

$$F(x_2) = 1 \times \frac{\sum((1, 1, 1) \times (0.6, 0.5, 0.7))}{\sum(1, 1, 1)} = 0.6$$

From the previous calculation x_1 will be selected because $F(x_1) > F(x_2)$. The selected result demonstrates that the sequence of x_1 is the most favorable in the current situation to be executed, considering the criteria and usage of this method. This may not be valid for other situations using the same model, where the decision values were changed in case they do not represent the actual characteristics. As an example, the energy efficiency of the equipment may vary during time, where the corresponding attribute must be recalculated and therefore influencing the decision made afterwards.

V. Conclusions and Future Work

In service-oriented automation systems, decision-making is an important task to support the conflict resolution, the exception handling and the reconfiguration and evolution processes. This paper introduces a novel approach for the process optimization in the orchestration of service-oriented automation systems, centered in the use of Petri nets to represents the work-plan associated to services, which will be interpreted and executed by orchestration engines. The information extracted from the Petri net models, such as the structure of the net and the transition-invariants, constitutes important knowledge that can be used to support the decision-making process. This knowledge is then combined with a flexible set of decision criteria, which can for instance consider production parameters but also energy efficiency issues. The application of this method permits the maintenance of the models for orchestration and also their evaluation for decisions. The dynamic optimization of processes can be reached in real-time, considering available parameters from devices and other resources.

Future work is related to test this method on compositional services (that will inherit the decision values), to the automatic definition of parameters by the devices (e.g. reading power consumption from the equipments) and also to study the variable values during operation.

Acknowledgment

The authors would like to thank the European Commission and the partners of the EU FP6 project “Service-Oriented Cross-layer infrastructure for Distributed smart Embedded devices” (SOCRADES) and the EU FP7 project “Cooperating Objects Network of Excellence” (CONET) for their support.

References

[1] E. Zeeb, A. Bobek, H. Bohn and F. Golatowski, “Service-Oriented Architectures for Embedded Systems Using Devices Profile for Web Services”, Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops, vol. 1, pp. 956-963, 2007.

[2] The Devices Profile for Web Service Specification. See http://schemas.xmlsoap.org/ws/2006/02/devprof/ (access date: January 2010)

[3] A. Bobek, E. Zeeb, H. Bohn, F. Golatowski and D. Timmermann, “Device and service templates for the Devices Profile for Web Services”, Proceedings of the 6th IEEE International Conference on Industrial Informatics, pp. 797-801, 2008.

[4] Q. Li, Y. Shu, C. Tao, X. Peng and H. Shi. “Service-Oriented Embedded Device Model in Industrial Automation”, Proceedings of the Second International Symposium on Intelligent Information Technology Application, vol. 1, pp. 525-529, 2008.

[5] N. Milanovic and M. Malek, “Current Solutions for Web Service Composition”, *IEEE Internet Computing*, vol. 8, n. 6, pp. 51-59, 2004.

[6] Web Services Business Process Execution Language. OASIS Standard. (http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html), 2007.

[7] R. Hamadi and B. Benatallah, “A Petri net-based model for web service composition”, Proceedings of the 14th Australasian Database Conference, pp. 191-200, 2003.

[8] X. Deng, Z. Lin, W. Cheng, R. Xiao, L. Fang, and L. Li, “Modeling Web Service Choreography and Orchestration with Colored Petri Nets”, Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, vol. 2, pp. 838-843, 2007.

[9] B. Medjahed, A. Bouguettaya and A. K. Elmagarmid, “Composing Web services on the Semantic Web”, In: *The VLDB Journal - The International Journal on Very Large Data Bases*, vol. 12, n. 4, Springer-Verlag New York, Inc., pp. 333-351, 2003.

[10] J. L. M. Lastra and M. Delamer, “Semantic Web Services in Factory Automation: Fundamental Insights and Research Roadmap”, *IEEE Transactions on Industrial Informatics*, vol. 2, n. 1, pp. 1-11, 2006.

[11] K. C. Thamboulidis, G. V. Koumoutsos and G. S. Doukas, “Semantic Web Services in the Development of Distributed Control and Automation Systems”, Proc. of the IEEE International Conference on Robotics and Automation, pp. 2940-2945, 2007.

[12] Z. Maamar, S. Kouadri and H. Yahyaoui, “A Web services Composition Approach based on Software Agents and Context”, Proceedings of the 2004 ACM Symposium on Applied Computing, ACM Press, New York, NY, USA, pp. 1619-1623, 2004.

[13] J. Pasley, “How BPEL and SOA are Changing Web Services Development”, *IEEE Internet Computing*, vol. 9, n. 3, pp. 60-67, 2005.

[14] J. Day and R. Deters, “Selecting the Best Web Service”, Proceedings of the 2004 Conference of the Centre for Advanced Studies on Collaborative research, IBM Press, pp. 293-307, 2004.

[15] S. Chaari, Y. Badr and F. Biennier, “Enhancing Web Service Selection by QoS-based Ontology and WS-policy”, Proceedings of the 2008 ACM symposium on Applied computing, ACM Press, New York, NY, USA, pp. 2426-2431, 2008.

[16] F. Jammes and H. Smit, “Service-oriented Architectures for Devices - the SIRENA View”, Proceedings of the 3rd IEEE International Conference on Industrial Informatics, pp. 140-147, 2005.

[17] M. Gerosa, A. Cannata and M. Taisch, “A Technology Roadmap on Service-Oriented Cross-layer Infrastructure for Distributed smart Embedded devices”, Proceedings of the I*PROMS Conference, 2008.

[18] Web Services Description Working Group (see http://www.w3.org/2002/ws/desc/).

[19] J. M. Mendes, F. Restivo, P. Leitão and A. W. Colombo, “Customizable Service-oriented Petri Net Controllers”, Proceedings of the 35th Annual Conference of the IEEE Industrial Electronics Society, 2009.

[20] J. M. Mendes, A. Bepperling, J. Pinto, P. Leitão, F. Restivo and A. W. Colombo, “Software Methodologies for the Engineering of Service-Oriented Industrial Automation: The Continuum Project”, Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference, pp. 452-459, 2009.

[21] T. Murata, “Petri nets: Properties, Analysis and Applications”, *IEEE*, vol. 77, pp. 541-580, 1989.

[22] C. Amer-Yahia, N. Zerhouni, A. E. Moudni and M. Ferney, “Some Subclasses of Petri nets and the Analysis of their Structural Properties: a New Approach”, *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 29, n. 2, pp. 164-172, 1999.