Proceedings of the 2008 IEEE
International Conference on Robotics and Biomimetics
Bangkok, Thailand, February 21 - 26, 2009

Semantic Web Services Framework for Manufacturing Industries

Andrei Lobov, Fernando Ubis Lopez, Vladimir Villaseñor Herrera, Juha Puttonen and Jose L. Martinez Lastra

Department of Production Engineering
Tampere University of Technology
P.O. Box 589, Tampere, Finland, 33101

{andrei.lobov, fernando.ubis, vladimir.villasenorherrera, juha.puttonen & jose.lastra}@tut.fi

***Abstract* – Application of Semantic Web Services (SWS) relies on ontologies which model existing explicit knowledge in problem domain. Adaptation of SWS to manufacturing industries rises a number of important questions among which is the separation of responsibilities between different vendors starting from system component manufacturers and ending with the system user. This paper gives a possible solution to settle these responsibilities and describes the architecture for the orchestration of semantic web services.**

Index Terms – Semantic Web Services, Factory Automation.

I. Introduction

Semantic Web Services (SWS) can be seen as self-contained and self-describing software components that can be discovered and invoked on the Web. These are seen as a next step in evolution of World Wide Web (WWW) [3]. A set of standards is defined by World Wide Web Consortium (W3C) [11] and Organization for Advancement of Structured Information Standards (OASIS) [12] in order to address different aspects of SWS application, some notable developments can be found in [8], [9] and [10].

Emerging as software engineering paradigm, SWS start to find their way in industry and in particular in the domain of factory automation [4], [7]. Different aspects of SWS application can be found in [1], [2], [3] and [5]. [1] contains a discussion on Artificial Intelligence (AI) and SWS stating that it was realized that "essential element of AI's knowledge-based paradigm is (…) casual relationship between a system's explicit knowledge representation and its (intelligent) behavior." The 'intelligence' here can be seen as an ability to extract implicit knowledge given explicit knowledge.

At the moment given the production line equipped with the robotic and manual workstations and transportation (e.g. conveyor) system, an engineer needs to obtain requirements telling what the entire production system should do. The requirements can come in a narrative form and/or be expressed in some supportive high-level languages, e.g. Gantt charts, flowcharts, etc. The required process is then implemented using dedicated Programmable Logic Controller (PLC) and robot programming languages. While being applicable for mass production, such a development process results in a production line that contains small room for reconfigurability. That is, on the arrival of new unforeseen product that customer wants to do with the line, the line should be reprogrammed to adapt new product demands.

SWS can be seen as a possible solution that enhances possibilities for production systems reconfigurability. SWS aim to detach a product from its manufacturing equipment through the processes required by the product and provided by the equipment. Therefore, the production line abstracted as a set of processes (services) it can provide allows to direct such system development based on the processes it provides instead of products it should manufacture. The idea is to shift from low-level programming of sequencing sensor readings and activations of actuators to the high-level programming of the pieces of equipments providing services.

Service description is done in Web Service Description Language (WSDL) [8]. The service invocations or service orchestration can be implemented by means of Business Process Execution Language (BPEL) [9]. The BPEL can allow expressions for cross-companies scenarios where service invocations are executed on different parties of the supply chain [6]. The BPEL originates from workflows; it is a graphical language that contains rectangular-shaped service invocation and diamond-shaped decision blocks. The structure of BPEL can be detailed down from higher level process involving different companies in supply chain, taking, for example, several days between different steps, to conveyor segment service invocation loading a pallet within a few seconds at the factory floor [13]. It should be noted however, that web services paradigm coming from so-called office automation should be applied with precautions at the factory floor due to inapplicability of simple rollback or recovery procedures available in the software world to the realm of the physical world.

Web Ontology Language (OWL) [10] is used to represent the knowledge available in the problem domain. The implicit knowledge can be derived by SPARQL Protocol and RDF Query Language (SPRQL) [14]. A set of ontologies have to be developed to allow inter-mapping processes, products and equipments. One can also define Semantic Web Services as the web services described in ontologies. A framework described in the paper defines the business chain for SWS and details architecture to allow application of SWS.

The rest of the paper is outlined as follows: second section details service orchestration framework, third section discusses SWS Framework. Fourth section gives an

application example. The conclusions are drawn in fifth section.

II. Service Orchestration Framework

The orchestration means the composition of the web services in order to provide more complex process (service). This composition can be supported by BPEL. Fig. 1 defines a service orchestration architecture having three main elements: orchestrator, service oriented middleware and decision support system (DSS). An orchestrator is a tool chain allowing discovery and definition of services in use. In addition it provides validation of service compositions. The role of the orchestrator is to check if the manufacturing of a given product is possible in *principle* given a set of available services. The orchestrator deploys an orchestration engine to the service oriented middleware.

The role of middleware is to outsource (execute) the functionality that may not be performed by the basic units in architecture (i.e. services – in this case). The middleware hosts a set of orchestration engines. The orchestration engine (OE) represents the needs of the product. The OE exists at run-time to fulfill the needs of a product. That is, it discovers available services, maps these to the existing devices and does the service invocation for product manufacturing. The OE can resolve some of basic conflicts such as mutual exclusion for the access by different products the same working area. However, some decision making is outsourced to DSS.

The DSS is a multi-agent system (MAS) attached to the same communication backbone as the orchestrator and the middleware. It is capable of performing functionality usually attributed to manufacturing execution systems (MES). That is, monitoring of throughput, load balancing, etc. The agents as the main building blocks of DSS coexist with services at runtime.

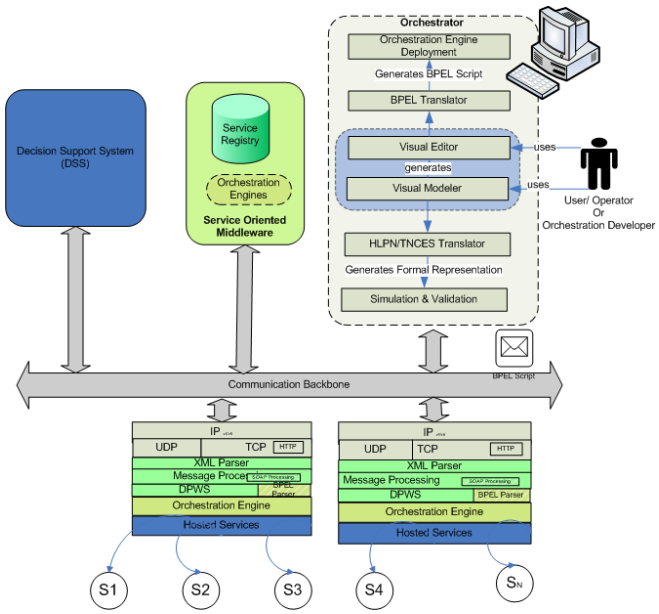

Fig. 1. Service Orchestration Framework

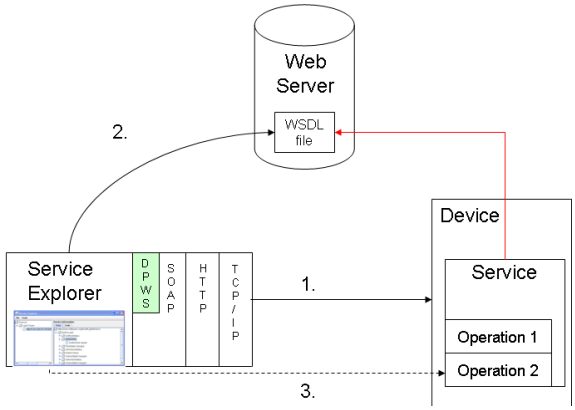

Fig. 2. Dynamic Service Invocation

One important feature of the systems following principles of Service Oriented Architecture (SOA) is the ability to be dynamically discovered and invoked. That is, the newly added device can inform on its arrival or be discovered by application of corresponding protocols. Device Profile for Web Services (DPWS) [15] is used to implement services in the production line. An example of the service could be a painting service of a robot which receives as an input an area that has to be painted or the transfer service for the conveyor segment. The orchestration of abovementioned services can be seen as a composite service that feeds the part into the working area of the robot for painting.

Principles of dynamic service invocation are highlighted in Fig. 2. First, the service is discovered by using WS-Discovery protocol of DPWS. A metadata of the service is uploaded in order to locate a WSDL file containing description of the service. A Service Explorer tool was developed to allow automatic uploads the WSDL files (2) for later service invocation (3). The tool is capable of automatic service orchestration by reading BPEL scripts performing the role of the orchestration engine. The orchestration engines can be deployed independently to solve given BPEL scripts at run-time. In certain circumstances, the OEs can request the DSS for a decision that can be provided in the form of another BPEL script.

The WSDL can give information how to invoke an operation on the device. However, it is solely the user's responsibility for possible consequences of service operation invocation. That is, an invoker must "know" what it invokes. At this level the semantic description of the device should be introduced to allow automated binding of a service. In addition to knowing the result of service invocation, one should be able to compose the services to address the product needs. Again a semantic description of product needs is required to be mapped with the semantic description of the equipment to check whether it is possible to fulfill the product needs. Four basic areas providing semantic description of a system and the product's needs were suggested in [7]. A Web Ontology Language (OWL) [10] is used to describe Process Taxonomy, Product Ontology, Equipment Ontology and Service Ontology.

Fig. 3 highlights the utilization of OWL files by orchestration engine in order to obtain the knowledge on currently available resources and the state of the world (e.g. number and location of products in the systems, operation status of the equipments, etc.).

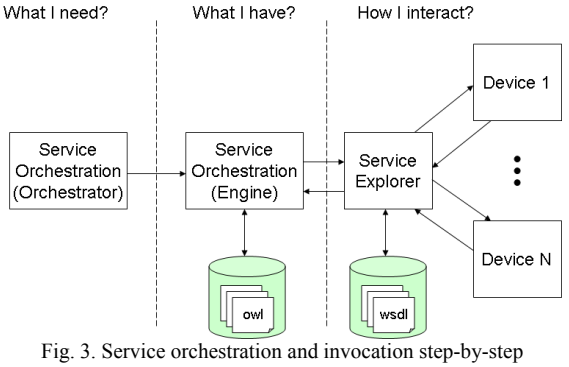

Fig. 3. Service orchestration and invocation step-by-step

As highlighted in the Fig. 3, a repository of OWLs files should exist to "intelligently" orchestrate the system. Given process taxonomy, the services provided by the equipments should be modeled in terms of the process taxonomy. The same is applicable for the products needs which have to be 'grounded' in the processes. OWL-S [16] – an upper ontology for web services – is a bridge between the WSDL representing physical devices available in the field and the equipments providing those modeled in OWL. Another option for including semantic information within a Web Service definition is by using Semantic Annotations for WSDL and XML schema (SAWSDL) [17]. SAWSDL is a W3C recommendation that defines extensibility attributes based on XML Schema, making possible to integrate semantic description to some of the WSDL components.

The overall process of product manufacturing from production system design to the ready made product being manufactured can be sketched as follows:

1. Obtaining the Process Taxonomy;
2. Production system design in terms of Process Taxonomy;
3. Definition of Equipment ontologies;
4. Product ontology definition;
5. Product manufacturing based on decisions made by querying and updating of Equipment and Product ontologies.

A question – who creates different OWL files? – is answered in the following section.

III. Semantic Web Services Framework

A set of tools are available to create ontologies and perform reasoning on those. A detailed report on different tool options can be found in [4]. We use Protégé tool [18] for ontologies development, which provides a number of plug-ins to support ontologies development. The Process Taxonomy, Product Ontology, Equipment Ontology and Service Ontology can be defined in Protégé. SPARQL queries can be used in order to derive implicit knowledge on the system. A semantic web framework for java (Jena) [19] provides an API allowing to manipulate ontologies and execute SPARQL queries on those.

In order to make the overall approach applicable, the key responsibility areas have to be defined. In order words the business chain has to be established. One can identify four main players for the production systems development and application business chain. These are device vendor, equipment vendor, system integrator and customer.

Here, the customer possesses the knowledge on the products that have to be manufactured. The device vendor manufactures and sells basic components such as sensors and actuators. The equipment vendor possesses an expertise on how to build the machines that can be used in production lines. The system integrator assembles the manufacturing facilities from the available equipments. The boundaries between different players can be 'blurred' in some cases, but in general the rationally here is straight forward: someone needs to provide sensors and actuators, someone needs to compose machines based on these and, finally, someone needs to integrate the production lines. The 'blurring' in the separation of responsibilities that exists at the moment may be attributed to the fact that the product indirectly dictates *what components* production line should have – how it should look like. The machine builder should have a deep knowledge on the problem domain of its equipment application. Although this situation seems to be natural and straightforward at the moment, it may contain certain constrains due to mindset of the machine builder. For instance, the basic operations of drilling and painting can be found in many sub-domains involving part manufacturing. However, due to narrow focus – e.g. electronics assembly – the machine builder can not see other business opportunities for its products (in this case – machines).

Fig. 4 highlights the role of four abovementioned players in expressing their expertise in terms of ontologies and WSDLs. Starting from bottom to up, the device vendor selling basic devices provides WSDL files. These can be for instance downloadable from its web site. The equipment vendor or machine builder has to define upper ontology in order to ground equipments built and the basic services these provide as a composition of devices. System integrators may elaborate equipment ontologies. For instance, the neighborhood information can be defined as the pieces of equipments are being assembled to the production line. In addition, the system integrators may need to express some specific information on the environment, which is related to the installation site of the production equipments. The customer introduces information on the product as product ontology.

The process taxonomy has to be derived by standard institution. The taxonomy allows bridging the product and the equipments that have to manufacture it. The ontologies are processed at runtime by orchestration engine that on one side queries these using SPARQL through Jena API and on the other dynamically invokes the services based on the WSDL information (Fig. 4.).

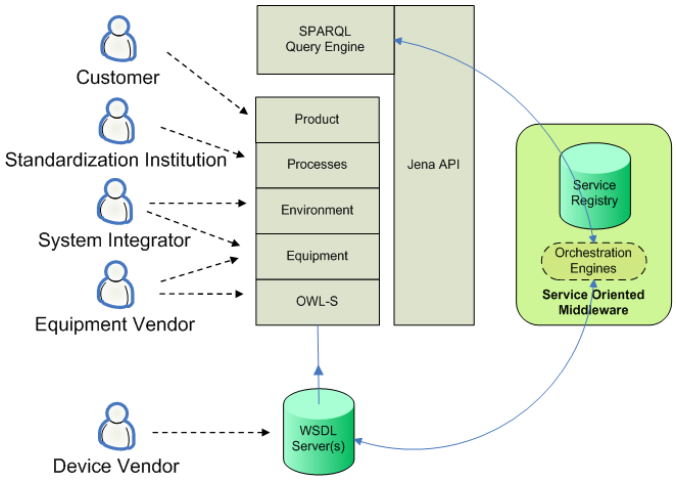

Fig. 4. Semantic Web Services Framework

IV. Application Example

Robotics and production systems ontologies have been defined in Protégé. The definition of an ontology starts with outlining main concepts. The concepts are then organized into hierarchy where the top element is a 'Thing' which properties are extended and inherited down to the sub-concepts or sub-classes. For instance the drive system of a robot can be electric, pneumatic, hydraulic or magnetic. Fig. 5 shows a part of robotics ontology, the ovals represent the concepts, the arcs represents the generalization relationship. That is, having most general class on the top (the 'Thing') the classes are specialized farther from the root adding some extra properties that make them differ from their siblings and their parents. Therefore, in given example, the 'DriveSystem' is just under the 'Thing' in class hierarchy and it is extended by four different drive systems. Like in object-oriented programming, ontologies can contain instances which may represent physical entities available in real world. The instance can be defined for any concept available in the ontology. In given example, five different robots were represented as different unique instances, among those is 'puma560' robot.

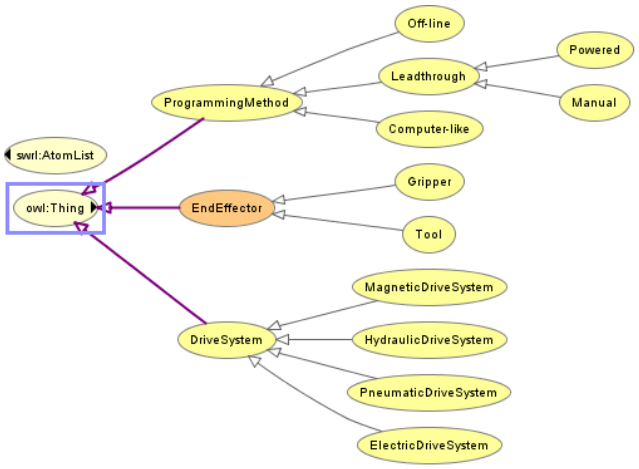

Fig. 5. Three concepts in robotics ontology

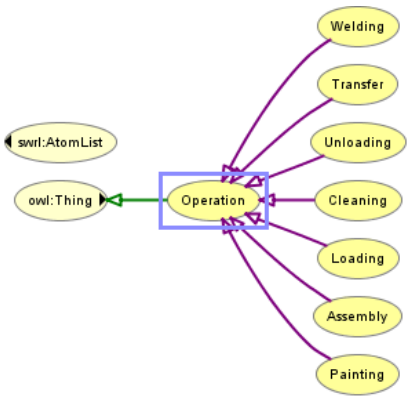

Fig. 6. Operations in robotics ontology

The robots are capable of providing certain operations. The operations shown in Fig. 6 are modeled in robotics ontology; these are Welding, Transfer, Unloading, Cleaning, Loading, Assembly, and Painting operations. The operations assigned to different robots describe the problem domain at hands – robotics. The link to the actual devices is done through OWL-S. An example of SPARQL query bridging these two domains is shown in Fig. 7. A spray gun attached to the 'puma560' robot provides a painting operation, while the robot provides 'Paint' service. The OWL-S plug-in in Protégé allows to link the 'Paint' service and 'puma560' robot through 'providedBy' predicate. Therefore if given product requires a painting service from the system a simple query can be used to obtain the result. As can be seen in Fig. 7, first the prefixes are defined in order to set up the possible locations of concepts, instances and predicates. 'SELECT' keyword filters the variables available in the query. '*' symbol denotes that all the variables (in this case '?object' variable) will be returned as a result. 'FROM' keyword highlights which document to search, while 'WHERE' keyword notifies what has to be searched and how. The query clauses are of the following format: "subject predicate object". An execution of the query returns the instance of the 'puma560' robot.

```
PREFIX robotics: <http://www.owl-ontologies.com/robotics.owl#>
PREFIX service: <http://www.daml.org/services/owl-s/1.2/Service.owl#>
SELECT *
FROM <http://www.owl-ontologies.com/robotics.owl>
WHERE
{ robotics:Paint service:providedBy ?object. }
```

Fig. 7. Query example for robotics ontology

```
PREFIX prod:<http://www.owl-ontologies.com/ProductionSystem.owl#>
SELECT ?operator ?workstation
WHERE { prod:pallet2 prod:containsProduct ?product.
?product prod:requiresOperation ?operation.
?operator prod:performsOperation ?operation.
?conveyor prod:containsPallet prod:pallet2.
?operator prod:worksAt ?workstation.
?workstation prod:hasConveyorFrame ?frame.
?frame prod:hasConveyor ?conv.
?conveyor prod:hasNeighbor ?conv.
}
```

Fig. 8. Query example for production system ontology

In comparison to rather trivial example provided in Fig. 7, Fig. 8 presents a query example on production system. Besides knowing that something can be done in the system in principle (e.g. there is a possibility to paint a product), one could be more interested to derive a solution on how a newly introduced product can be manufactured in the system. Having manual and robotic workstations in production system one may be interested in finding a path for product to reach the locations where the product needs can be (partially) fulfilled. This scenario may require development of more complex queries similar to the one shown in Fig. 8. The query in Fig. 8 is not concerned with the operations the product (prod:pallet2) is requiring, but rather it allows finding an operator fulfilling the needs of the product. Given query, if succeeds, besides finding the operator also confirms that there exists a *path* in the system to the *workstation* capable to fulfill product *needs*. 'SELECT' keyword can be used in order to clarify the required output. The predicates can be made transitive which significantly simplifies the query format. For example, 'hasNeighbor' predicate is defined for the conveyor segment. Being transitive, the predicate allows testing if pallet containing a product can reach desired workstation by having a single line in the query: '?conveyor prod:hasNeighbor ?conv.'

During system runtime, the ontologies have to be kept up to date depending on the situation in the system. The orchestration engines equipped with the SPARQL query engines bear this responsibility.

V. Conclusions

The Semantics Web Services Framework has been discussed in the paper. The goal of the framework is to assess the role of each element of the business chain in development and application of service-enabled production systems. An example based on robotics and production systems ontologies was given to highlight basic ideas of ontologies development and reasoning possibilities.

The application of SOA principles allows implementing loosely-coupled solutions for production systems aiming at plug & play and reconfigurability capabilities for this class of systems. From the end-user perspective, the goal is to shift from hard coding of the "clients" (control applications for the equipments) to the run-time discovery and usage of the services (abstracting the equipments).

Acknowledgment

The research is supported by SOCRADES research project under the EU's 6th Framework Programme (http://www.socrades.eu).

References

[1] M. d'Aquin, E. Motta, M. Sabou, S. Angeletou. L. Gridinoc, V. Lopez and D. Guidi, "Toward a New Generation of Semantic Web Applications," *IEEE Intelligent Systems*, vol. 23, pp. 20-28, May-June 2008.

[2] M. Hepp, "Semantic Web and Semantic Web Services," *IEEE Intelligent Computing*, vol. 10, no. 2, pp. 85-88, March-April 2006.

[3] R. G. Pereira and M.M. Freire, "SWedt: A Semantic Web Editor Integrating Ontologies and Semantic Annotations with Resource Description Framework," *In Proc. of IEEE Advance International Conference on Internet and Web Applications and Services* (AICT/ICIW'06), 2006

[4] J. L. Martinez Lastra, I. M. Delamer, F. Ubis Lopez, *Domain Ontologies for Reasoning Machines in Factory Automation*, Tampere University of Technology, Institute of Production Engineering, Report 71, ISBN 978-952-15-1522-4, Tampere 2007

[5] M. P. Papazoglou, P. Traverso, S. Dustdar and F. Leymann, "Service-oriented computing: state of the art and research challenges", *IEEE Computer*, vol. 40, no 40, pp. 38-45, November 2007

[6] A. Bartoli, R. Jimenez-Peris, B. Kemme, C. Pautasso, S. Patarin, S. Wheater, and S. Woodman, "The adapt framework for adaptable and composable web services". *IEEE Distributed Systems Online*, September 2005.

[7] I. Delamer, and J. L. Martinez Lastra, "Loosely-coupled automation systems using device-level SOA", *In. Proc. of 5th IEEE International Conf. on Industrial Informatics (INDIN'07)*, vol. 2, pp. 743-747, July 2007

[8] Web Services Description Language (WSDL) Version 1.1, available online (August 2008) at: http://www.w3.org/TR/wsdl

[9] Web Services Business Process Execution Language Version 2.0, available online (August 2008) at: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

[10] S. Bechhofer, et al., "OWL Web Ontology Language Reference", W3C, February 2004, available online (August 2008) at: http://www.w3.org/TR/owl-ref/

[11] World Wide Web Consortium (W3C), http://www.w3c.org

[12] Organization for Advancement of Structured Information Standards (OASIS), http://www.oasis-open.org

[13] J. Puttonen, A. Lobov and J. L. Martinez Lastra, "An Application of BPEL for Service Orchestration in an Industrial Environment," *In Proc. of 13th IEEE International Conference on Emerging Technologies and Factory Automation*, September 2008, in press

[14] W3C, "SPARQL Query Language for RDF", W3C Recommendation, 15.01.2008, available online (August 2008) at: http://www.w3.org/TR/rdf-sparql-query/

[15] Device Profile for Web Services (DPWS), available online (August 2008) at: http://schemas.xmlsoap.org/ws/2006/02/devprof/

[16] W3C, "OWL-S: Semantic Markup for Web Services", W3C Member submission, 22.11.2004, available online (August 2008) at: http://www.w3.org/Submission/OWL-S/

[17] W3C, "Semantic Annotations for WSDL Working Group", available online (August 2008) at: http://www.w3.org/2002/ws/sawsdl/

[18] Stanford Center for Biomedical Informatics Research, "The Protégé Ontology Editor and Knowledge Acquisition System", available online (August 2008) at: http://protege.stanford.edu/

[19] Jena, "Jena – A Semantic Web Framework for Java", available online (August 2008) at: http://jena.sourceforge.net/