

## A service-oriented architecture and its ICT-infrastructure to support eco-efficiency performance monitoring in manufacturing enterprises

Priscila Angulo, Claudia Cristina Guzmán, Guillermo Jiménez\* and David Romero\*

*Center for Innovation in Design and Technology, Tecnológico de Monterrey, Monterrey, México*

*(Received 5 May 2015; accepted 18 January 2016)*

Energy consumption and contaminant emissions to the ecosphere are two of the main environmental concerns nowadays in the industrial landscape. Both environmental performance indicators are currently being observed very closely by governments and society for their long envisioned impact on our planet's future. Moreover, the environmental performance of manufacturing enterprises should not be studied in isolation, since production contributes significantly to the gross development product of all nations, so enterprises' economic productivity and environmental performance, known as eco-efficiency, should be considered intertwined. Therefore, at the same time that production environmental footprint is considered, production economic aspects are also part of the green equation for achieving eco-efficient manufacturing enterprises. Furthermore, as enterprise information systems and manufacturing technologies evolve, their usefulness should be considered to aid factories to become more eco-efficient and reduce their energy consumption and emissions footprints in a competitive way. The Research & Technological Development work reported in this paper is part of the Factory ECO-MATION EU-FP7 project, which aims to create eco-efficiency monitoring solutions for eco-factories. The Factory ECO-MATION architecture integrates data from environmental sensors, production information and energy consumption records in order to allow their simultaneous monitoring, so that the production planning and control parameters can be adjusted to keep all production goals at a reasonable scale while protecting the environment. This paper describes an ICT-Infrastructure based on a service-oriented architecture (SOA) capable of integrating production and environmental information from diverse data sources in order to support the eco-efficiency performance monitoring of manufacturing enterprises.

**Keywords:** eco-efficiency; energy consumption; emissions footprints; production; SOA

### 1. Introduction

Manufacturing contributes significantly to Gross National Product (GNP) and employs a significant task-force in industrialised countries (Bi, Xu, and Wang 2014). A good reference describing how manufacturing enterprises contribute to the GNP in industrialised countries can be found in Cirani et al. (2014). However, while these enterprises contribute to the economic development of nations, they also affect the environment with the depletion of production resources such as the consumption of non-renewable energy sources and scarce raw materials (Shao et al. 2012). Although there are many technologies and techniques for achieving eco-friendly production operations, it is clear that any eco-efficiency strategy should simultaneously consider three types of efficiency in order to allow a manufacturing enterprise to pursue it in a viable way: (1) economic efficiency, thus enterprises can financially survive; (2) resources efficiency, necessary for keeping operations competitive at a low-cost determined by energy and raw materials consumption; and (3) emissions efficiency, since it is very important to maintain contaminant emissions at a low-rate due to environmental regulations and responsibility.

Information and Communication Technologies (ICTs), especially enterprise information systems (e.g. production planning and control systems), have helped to improve information management and therefore decision-making at different enterprise levels, including the shop-floor, and thus leading to higher levels of productivity (Ren, Zhang, and Tao 2010). However, most of the Research & Technological Development (RTD) conducted at the shop-floor has been addressing only production information management with software architectures (Xie and Gui 2012) and frameworks (Shao et al. 2012) targeting traditional manufacturing information integration for decision-making, disregarding in many cases the environmental information related to manufacturing, which nowadays has become extremely relevant for environmental compliance, auditing and certification (Amrina & Yusof, 2011; Koho, Torvinen, and Romiguer 2011).

The RTD work reported in this paper is part of the Factory ECO-MATION (Factory ECO-friendly and energy efficient technologies and adaptive autoMATION solutions) EU-FP7 project, which aims to create eco-efficiency monitoring solutions for eco-friendly factories. In order to accomplish this, different sensing and communication network technologies were used to gather data from

---

\*Corresponding authors. Email: (Guillermo Jiménez) [guillermo.jimenez@itesm.mx](mailto:guillermo.jimenez@itesm.mx); (David Romero) [david.romero.diaz@gmail.com](mailto:david.romero.diaz@gmail.com)

environmental sensors, production information and energy consumption records in order to allow their simultaneous monitoring, so the production planning and control parameters can be adjusted to keep all production goals at a reasonable scale while protecting the environment. Moreover, within the Factory ECO-MATION architecture, sensors are responsible for gathering real-time data directly from environment, while production information and energy consumption records are gathered from external applications using Web services.

Furthermore, one of the main goals of this RTD work is to develop a service-oriented architecture (SOA) and its ICT-infrastructure that can be easily deployed in different manufacturing enterprises, so that the integrated data can be used by a Decision Support System (DSS), which will be able to analyse it in order to suggest how the production planning and control parameters can balance production productivity goals and expected environmental production performance. The integrated data can also be used by other enterprise information systems such as Environmental Management Systems (EMS) for not only reporting environmental performance, but also eco-efficiency<sup>1</sup> performance.

The paper has been organised as follows: [Section 2](#) briefly presents related work on data integration; [Section 3](#) concentrates on the importance of eco-efficiency achievement by manufacturing enterprises; [Section 4](#) describes the functional requirements that justify the design of an SOA and its ICT-infrastructure; [Section 5](#) defines the requirements of a middleware to handle the identified functional requirements; [Section 6](#) analyses different alternative services technologies based on their complexity and specific needs from the Factory ECO-MATION architecture; [Section 7](#) presents the architectural solution for virtualising the data sources for eco-efficiency performance indicators measurement at a manufacturing enterprise coming from its enterprise information systems and sensor networks; [Section 8](#) provides the technical implementation details of the SOA and its ICT-infrastructure; [Section 9](#) describes the resulting functional prototype and the challenges in its development; and [Section 10](#) offers conclusions and future work.

## 2. Related work on data integration

Data integration challenge has been addressed in several research works and projects. For instance, Hoschek and McCance (2001) described in their work the Spitfire project efforts to address complexity, high performance and interoperability in data-grids. The Spitfire project provides a grid-enabled relational database middleware as a solution to reduce complexity and deliver high performance in data integration efforts. Furthermore, Rodríguez-Martínez and Roussopoulos (2000) in their work offer a self-extensible database middleware system for distributed data

sources called MOCHA, which interconnects over computer networks. MOCHA is designed to scale to large environments and it is based on the idea that some of the user-defined functionality in the system should be deployed by the middleware itself. It uses Java code for implementing advanced types of query operators to remote data sources and executing them remotely. In the case of Kim et al. (2014), their work describes the design and implementation of an information and communication system for managing a livestock farm. The system consists of sensors, sensor manager, database, servers and a user interface for an environmentally optimal breeding livestock farm. A common approach in these projects is the use of XML-based Web services to deliver the data obtained from the sensors. One difference between these approaches and the approach proposed later in this paper is the use of RESTful-based Web services.

## 3. Eco-efficiency importance for manufacturing enterprises

Being the manufacturing sector one of the main consumers of energy and accountable for a large percentage of contaminant emissions in the industrial landscape, and at the same time one of the biggest contributors to the GNP and employability rates in nations, it requires a special attention and assistance in finding eco-efficient ways for balancing its productivity and environmental performance. It is important then that manufacturing enterprises harmonise production efficiency with energy consumption and emissions (Amrina & Yusof, 2011), since it is no longer enough for an enterprise to be economically viable, neither to control their energy consumption to set it at the lowest possible level as well as their contaminant emissions, it is about to achieve the optimal production performance with the right balance between energy consumption, emissions and productivity (Zhang et al. 2011; Programme 2014).

## 4. SOA and its ICT-infrastructure functional requirements

Modern enterprises are heavily based on enterprise information systems that leverage information and communication technologies. These technologies enable intra- and inter-organisational cooperation and collaboration while supporting business processes and strategic decisions. Usually, this kind of support originates from data that constitute the building block of the analysis of each process. These data are also fundamental in driving and monitoring the execution of some enterprise business processes or allowing the completion of some others. For these reasons, data must be collected, filtered, merged and finally made available as information for the decision-makers (or to trusted users) through enterprise information systems (Xu et al. 2014).

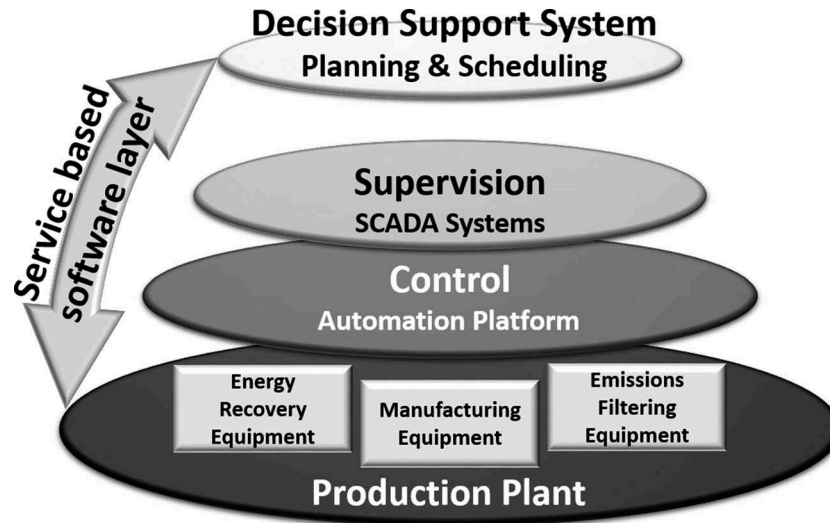


Figure 1. Factory ECO-MATION architecture.

Among the various sources of data related to all business processes of an enterprise, the physical environment can be considered as one of the most important ones (Ren, Zhang, and Tao 2010). In fact, the production process has usually strict interactions with the environment, which can have a significant impact on the quality of the final product, both directly, in case of outdoor production, and indirectly, by affecting the correct functioning of the factory plant. Nowadays, available technologies for such sensing capabilities are characterised by some distinctive features that must be carefully taken into account. In particular, the ICT infrastructure within an enterprise can be characterised not only by a huge amount of different kinds of software running over different operating systems, but also by a lot of different hardware platforms. From business level to shop-floor level, different data exchange standards are used because various types of data have to be communicated. At shop-floor level, sensed data have to be gathered by consumer applications through, generally, a proprietary communication protocol while at business level applications data exchange is much more standardised. Then, at shop-floor level, sensors are usually of very small size and, sometimes have few resources such as: limited amount of power (e.g. they are typically battery-powered with little or no capability of harvesting power), small communication bandwidth and reduced computational capabilities. They embed a very simple operating system that provides essential features as: event-driven computation and multitasking, basic interaction with other sensors, power control, and radio communication. At business and shop-floor levels, applications that gather sensed data or use it are different among them in terms of access levels, communication protocols and data availability constraints. As a consequence, because of these constraints, operations within sensor networks have to be performed by directly exploiting the underlying hardware components and/or the embedded

operating system. This complexity is certainly unacceptable for non-trivial applications, which must preferably be designed to be flexible, reusable and reliable. Thus, the necessity of a middle layer of software arises, i.e. a middleware, which should be laid between the embedded operating systems and the applications; a software layer which hides the complexity of low-level pervasive technologies while helping the higher software layers gather heterogeneous real-time data coming from the production (see Figure 1).

### 5. Middleware layer functional requirements

Following the presented objective for the Factory ECO-MATION sensing and monitoring platform, a software layer should provide the following components, which are considered the guidelines for the functional requirements definition:

- *Programming abstractions* that provide high-level programming interfaces with different abstraction levels and programming paradigms.
- *Standardised system services* that are exposed through abstraction interfaces and can provide various management functions.
- *Run-time environments* that can extend the sensors embedded operating system in supporting and coordinating multiple applications to be concurrently run over the developed platform.
- *Quality-of-Service (QoS) mechanisms* that allow the adaptive and efficient utilisation of system resources.

Moreover, the software layer should obey to certain design principles that, as it can be argued, on the one hand should address the heterogeneous sensor networks distinctive

features, and on the other hand should meet the application needs. Especially, the following key principles have to be considered when designing the software layer:

- *Data orientation* – within a sensor network, the main focus is not on the node that produces data but on the data itself, i.e. the application is more interested on data than on the identity of the data source. For this reason, a software layer should support this principle of data-centric networking by means of proper techniques of data extrapolation or specific routing and querying within the network.
- *Energy awareness* – Energy management is another key issue for a sensor network, especially for wireless devices where sensor nodes are typically powered by batteries that could not be easily replaced; particularly when many nodes are involved and distributed over wide and sometimes impervious areas. Therefore, in order to increase the network lifetime, the software layer should be capable of managing properly the sensor hardware resources, especially communication bandwidth and processing speed.
- *Quality-of-Service support* – A sensor network should always provide the performance level required by the applications, i.e. the required QoS level. Unfortunately, sensor networks are subject to many status variations (e.g. energy loss, hardware failures, etc.) during their lifetime, and thus the required QoS level could not be provided if proper in-time actions are not taken into account. The software layer for sensor networks should provide QoS support by allowing adaptive changes in the network, affecting determined QoS properties like event notification reliability or sensing information accuracy.
- *In-network processing* – Sensor nodes constitute elements of a distributed network in which transmitted bits regard not only to the collected data but also to the information about the network itself. For example, in-network processing should be supported by the software layer, and data aggregation and data compression are two techniques that are important in this case because they allow reducing data transmission in the network, respectively, by condensing it or by correlating it. The classic example of data aggregation is the distributed evaluation of a maximum value.
- *Scalability and Robustness* – are correlated from a software layer point of view, as they both relate to the performance of the network as a function of its size. On the one hand, the software layer should support scalability by continuing to offer a certain performance level even when the number of network nodes grows, and on the other hand the

software layer should support robustness by tolerating node failures that can reduce network size or change network topology.

- *Reconfigurability and Maintainability* – A sensor network is usually deployed to be long time living, as the coverage area can be very wide and impervious, and sensor nodes could not be easily reachable. However, after the initial deployment, new necessities may arise and applications may need to change tasks of some nodes or assign them new tasks to perform. For these reasons, the software layer should support reconfigurability and maintainability of the network with minimal or null manual intervention.
- *Heterogeneity* – Sensor nodes can differ from each other due to their hardware features (such as battery capacity, transmitting power, processing speed, sensing capability, etc.) at deployment time or while the system evolves, due to different tasks each node has to perform. The software layer should support this heterogeneity by adopting a proper task dispatching policy based on the state of nodes.
- *Real-world awareness* – Sensor networks are used to monitor phenomena that happen in the real world, subjected to time and space laws. In many cases, applications could be interested in knowing when and where a certain datum has been collected or an event happened, the software layer should support them in this sense.

NOTE: In some cases, these principles could affect each other; in that case, appropriate solutions that balance all the aspects need to be found, keeping in mind the specific application requirements.

Having in mind the components and the above-mentioned features, and within the context of Factory ECO-MATION sensing and monitoring requirements, the mentioned software layer has to be conceived. For such aim, the adoption of SOA approach, in which all systems functionalities are deployed as services, was chosen. These services can manage requests from different high-level applications and communicate to achieve the required level of performance. For this reason, two types of interfaces have to be defined. The first is to enable the connection with applications and the second one is to allow the connection with the low-level sensor layer.

Other important requirement is the appropriate set-up stage of the type of services. In the set-up stage it performs an analysis of compatibility, reliability and quality of service in order to avoid undesired delays during operation stage.

Several technologies and frameworks were analysed in order to identify the most appropriate one for fulfilling the Factory ECO-MATION sensing and monitoring platform objective. The following section presents an analysis of those technologies.

## 6. Middleware supporting technologies background

In order to understand why Web services are one of the better choices for achieving the Factory ECO-MATION goals of interoperability and flexibility, it is necessary to review a little history of the evolution in information sharing and data exchange technologies. The trend has been guided by the necessity of information and data integration across dissimilar platforms. Each platform uses different data representations, thus the need for standardisation has been the major driver for different techniques and technologies (Linthicum 2003).

Among the several techniques and technologies that have been proposed along the time, the most significant ones are the following, chronologically organised, from older to newer:

- *Text file sharing* – Every platform uses a different internal representation for data, so it makes difficult the direct data sharing among platforms. The only data representation that is standard across platforms is ASCII text. This is why, in the beginning, to port data from one platform to another, the data were translated to some structured representation in an ASCII text file in a magnetic media of some sort. Then the magnetic media was inserted in the target platform and translated to that platform's internal representation. That process was necessary for every chunk of data that needed to be ported from one platform to another.
- *Electronic Data Interchange (EDI)* – The first version of a middleware to automatically move data from one platform to another was EDI, which was used a proprietary and very cryptic representation of data, and only those platforms for which the EDI middleware existed were able to use EDI. One constrain in EDI was that the only interaction among platforms was for file interchange, other types of interactions were not possible (Qi 2001).
- *Common Object Request Broker Architecture (CORBA)* – CORBA was the proposal of a specification from several of the bigger technology companies for object-oriented based applications (e.g. C++, Java, etc.). In CORBA, the broker consisted in three components: a proxy, an adapter and a communication layer. In CORBA, the two objects requiring interaction were named the client and the server. When a CORBA client sends a message to a server, the proxy at the client side translates the message to an internal representation, and then the server uses the adapter to translate the message to its internal representation, performs the necessary tasks, and sends the result back to the client. A main contribution that CORBA made was that clients and servers could interchange many messages, with every message directed to a specific server. However, besides allowing interoperability, CORBA also included security, transaction management, naming services and many other specifications, thus making its complete implementation very difficult. These reasons caused the reduced use of CORBA for application integration (Feinberg 1950).
- *COM+ and DCOM* – The Communication Object Model (COM) and its distributed versions were proposed by Microsoft (Davis and Zhang 2002). Although in principle they would be implemented in different platforms, in reality only Windows implemented them. Microsoft Windows being the dominant platform in the IT marketplace, COM+ and DCOM were broadly used in business and manufacturing integration. However, both COM+ and DCOM were complex and evolving very rapidly with many changes in their specifications, making difficult their implementation and keeping updated to their last version.
- *Java RMI* – Java is a platform-independent programming language, mainly based on packages or frameworks for its extensibility. Remote Method Invocation (RMI) was the Java proposal for remote object interaction, and an RMI package was added to the Java programming language (Burke 2009). RMI is very flexible, but it is limited to interoperability between Java only applications. Because many applications were implemented in C++ and other object-oriented programming languages, using RMI was a limited solution.
- *Web Services* – The Web services SOAP standard was proposed by many of the main IT enterprises, among which was Microsoft itself (Davis 2012). The proposed standard specifies that all necessary interactions among clients and servers would be in an ASCII text representation, in a specific format: XML. Because, as explained before, ASCII text is the basic representation that every platform understands. Web services were the most flexible interoperability proposal at that time. It was now possible that a client running in any platform could interchange messages with a server running in any other platform. A server could implement and deploy many services for clients to call (consume). At the beginning, it was difficult to describe and consume Web services because many XML codes were necessary to specify services. These problems were highly reduced by a combination of programming language compilers and the development of environmental enhancements. Currently, the code intended to be consumed (called from clients) is annotated by predefined syntax, and the integrated development environment (IDE)

generates the necessary XML code to deploy it as a service.

- *Internet Communications Engine (ICE)* – is a distributed computing platform designed with interoperability and scalability in mind (ZeroC 2003). ICE can be deployed in a large number of operating systems (Windows, Linux, OS X, Windows RT, Solaris, etc.) and programming languages (C++, C#, Java, Python, Objective-C, Ruby, PHP and ActionScript). The main design goals of ICE are to:
  - Provide an object-oriented middleware platform suitable for use in heterogeneous environments.
  - Provide a full set of features that support development of realistic distributed applications for a wide variety of domains.
  - Avoid unnecessary complexity, making the platform easy to learn and use.
  - Provide an implementation that is efficient in network bandwidth, memory use and CPU overhead.
  - Provide an implementation that has built-in security, making it suitable for use over insecure public networks.

How the previous approaches and technologies support the needs for the Factory ECO-MATION platform are summarised in Table 1.

Among the main concerns in the Factory ECO-MATION architecture are flexibility and interoperability, but also simplicity should be added as an important factor. Considering the requirements from the Factory ECO-MATION sensing and monitoring platform, and according to Table 1, it could be seen that Web Services are the best choice. The following subsection describes the main characteristics of architectures based on services.

### 6.1. Service-oriented architecture evolution

The SOA is a reference architecture in which functional resources are exposed through interfaces defining the available services, including their name and input and output parameters. How services are described, and the mechanisms for their access, is determined by a specific

middleware. Along the time, several technologies have been proposed for supporting the services concept with every one using a different approach. These technologies differ mainly in flexibility and supported software platforms (e.g. programming languages and operating systems) (Murer and Szyperski 2002; Linthicum 2003). It is important to understand them, so a well-founded approach could be defined.

As described above, the evolution of middleware (or services) technologies can be divided into two stages. The first generation of middleware was aimed at integrating legacy systems in a single platform (e.g. COM, DCOM), and among different platforms (e.g. CORBA). Meanwhile, the second generation was aimed to create a market of services available to everyone with access to the Internet. While the first generation was developed mainly based on interoperability concerns between and among enterprises, the second generation expanded the possibility to be available to any individual with Internet access, or more specifically, through the Web (e.g. the HTTP protocol). Data communication in the first generation was mainly represented in binary content. The second generation allowed multimedia content to be shared among applications.

The first generation was addressed to software applications communication, using specific APIs (Application Programming Interfaces) particular to every implementation. The second generation represented the democratisation of communication, because they allowed access to information from a Web browser or software library supporting the HTTP protocol (in which the Web is based). Borrowing the concept of services from the first generation, and with the possibility to communicate through the HTTP protocol (the protocol of the Web), the second generation was generally recognised as Web Services (e.g. software services accessible through the Web protocol).

The first implementation of the Web Services concept offered several capabilities through SOAP payload and WSDL description (de Oliveira et al. 2013):

- *Web services as a standard protocol* – The main interoperability attraction on the second generation

Table 1. Middleware technologies and approaches summary.

	Text File	EDI	CORBA	COM+	Java RMI	Web Services	ICE
Flexibility	+	-	-	+-	+-	+	-
Messaging	-	-	+	-	-	+-	-
Performance	N/A	+	+	+	+	+-	+
Availability	N/A	+	+	+	+	+	+
Simplicity	+	-	-	-	-	+-	-
Interoperability	+	-	+	-	+-	+	+-

was the Web protocol for information communication, and its universal support by the Internet.

- *Web services as proxies to information sources* – A simple way for communication was the possibility of Web browsers to communicate through proxies to consume services, which could communicate to local or remote applications.
- *Web services choreography* – Many applications require consuming more than a single service and services consumption require specifying an order in which they should be consumed. Web services supported that functionality through BPEL (Business Process Execution Language), which allows specifying the order in which a set of services should be accessed.
- *Web service negotiation* – With the possibility of multiple Web services implementing similar functionality, but with different characteristics (e.g., cost, security, etc.), it was necessary to be able to select among alternatives. Service negotiation addressed the possibility to select the most appropriate alternative.

The concept of Web services gained momentum mainly because of their simplicity for interoperability. Along the time, it was observed that in many software applications, mainly for application integration inside a single enterprise or even among enterprises, service negotiation was never required. These observations were one of the main reasons why a new reference architecture for Web services description and consumption was proposed by Roy Fielding (2000), known as the RESTful approach for Web services. The RESTful approach proposal has become an adequate alternative mainly because how enterprises were using Web services. However, as the simplicity and characteristics of RESTful Web services are better understood, its use has grown enormously, becoming the main way in which companies such as, e.g., Google, Amazon, Dropbox, Microsoft, etc. are implementing their Software-as-a-Service (SaaS) offers for their customers (Lomotey and Deters 2013). Besides, REST services are processed up to three times faster than SOAP services (Aihkisalo and Paaso 2012).

Now, with two ways on how Web services could be implemented, it is important that requirements should be carefully analysed to determine the most appropriate approach.

## 6.2 SOAP vs. RESTful for the service layer implementation

This subsection analyses SOAP and RESTful approaches using several criteria: easy-of-deployment, performance, security, simplicity and adaptability. Table 2 shows the

differences between a SOAP-based approach and a REST-based approach.

The requirements of the Factory ECO-MATION architecture and infrastructure are different from those described in Pautasso, Zimmermann, and Leymann (2008). However, the evidence gathered from other research results conveys a similar result: RESTful services provide a software architecture paradigm appropriate to comply with the Factory ECO-MATION architecture expectations. When Web services will be used inside an enterprise, as in this case, service negotiation would become of lesser importance, this being one of the main reasons why RESTful approach was selected for Web services implementation. Another important thing not addressed in this paper is that in order to simplify the architecture of the Services Provisioning Layer, something that should be considered is how services would be modelled (Liskin, Singer, and Schneider 2012). This modelling is necessary before services implementation.

## 7. Architectural solution for virtualising the eco-efficiency performance indicators

This section describes how different data sources in manufacturing enterprises can be integrated to create a single database to support the measurement of eco-efficiency performance indicators. Figure 2 shows a scenario detailing consumers and data sources. There are three data sources: production management information systems, energy consumption sensors and environmental sensors. Pollution records are obtained from a set of wireless sensors distributed along the shop-floor that take measurements from liquids and the air. Energy consumption and production data are obtained from applications already operating in a manufacturing enterprise (e.g. production planning and control systems).

All those data are available to consumer applications through the virtualisation layer, which hides the data sources by defining and implementing a consistent way of communication from applications to data sources. Communication between virtualisation layer and production and energy recording applications is performed through different protocols and data formats, as shown in Figure 3.

The sensor layer records the sensors outputs in a database server which is directly accessible through the virtualisation layer. The payload between consumer applications and the virtualisation layer can be delivered in JSON or XML formats, as specified by the consumer applications, as depicted in Figure 4.

One important requirement is that the ICT-infrastructure should be easily deployed in different manufacturing enterprises. The following section describes how this requirement is being dealt with.

Table 2. SOAP vs. RESTful.

SOAP	RESTful
<p><b>Easy-of-deployment</b>                      At the server side, when using WS* Web services (SOAP, WSDL, WS-Addressing, WS-Security), whenever a service is relocated its WSDL needs to be regenerated in order to include the new end points. Similarly, SOAP schemas and deployment descriptors need to be modified (this is so, because all mentioned files bring with them the URL of the server in which they will be deployed). At the client side (the Factory ECO-MATION applications), the new WSDL specifications should be used to recreate (or complete the definition of) the clients.</p>	<p>In RESTful services, the state is dynamically maintained at the client side. The state could be changed by new values received from the server. Among the values there could be URLs to resources that can be accessed from the current Web page. This means that the server keeps a structured organisation of the resources (only the root changes from one server to another), making simple the deployment of RESTful services.</p>
<p><b>Performance</b>                      According to Mulligan and Gracanic (2009), SOAP encloses each message payload within an additional SOAP envelope (set of XML tags) and adds a few SOAP-related headers to the outbound HTTP packet. This and this alone contributes to the added bloated in SOAP packets. Besides, the servers incur in additional latency because of the need to parse the additional payload.</p>	<p>The Factory ECO-MATION infrastructure will be mainly performing GET operations. Besides the ID of the KPI or sensor, requests will not include any XML code (Mulligan and Gracanic 2009), thus the server will not incur in any additional delay to dispatch them.</p>
<p><b>Security</b>                      The SOAP specification includes both authentication and authorisation mechanisms for securing Web services. This is necessary when the services will be available for their consumption from outside the enterprise, which will be the case for the Factory ECO-MATION infrastructure, when portable devices will have access to the services from all abroad.</p>	<p>As mentioned in Burke (2009), RESTful services also define authentication and authorisation mechanisms for securing access to the services. Besides sensitive data could be protected with cryptographic services like SSL. The Web defines the HTTPS protocol to leverage SSL and encryption. Thus RESTful services may use already available HTTP protocols and encryption mechanisms and implementations.</p>
<p><b>Simplicity</b>                      The WS*services stack requires WSDL, XML-schema, UDDI and XSL (Pautasso, Zimmermann, and Leymann 2008). Although such specifications have been around for some time, it is not so simple to implement all of them for every service. This adds to the complexity of WS*services.</p>	<p>The advantage of RESTful services proposal remains in its simplicity, because the only mechanisms that need to be understood are those of the Web (HTTP particularly), and those have been among us for more than 20 years (Pautasso, Zimmermann, and Leymann 2008). Comparing the complexity of Web service interface design, we can conclude that RESTful appears to be simpler.</p>
<p><b>Adaptability (how easy it is to add services)</b>                      Every SOAP service requires creating a WSDL and perhaps a WS-schema for the service. If the development environment does not automatically generate them, it would also be necessary to manually modify the deployment file to include every new service.</p>	<p>In this case, the developer will need to analyse the services structure to decide where the implementation is to be stored, and that is it.</p>

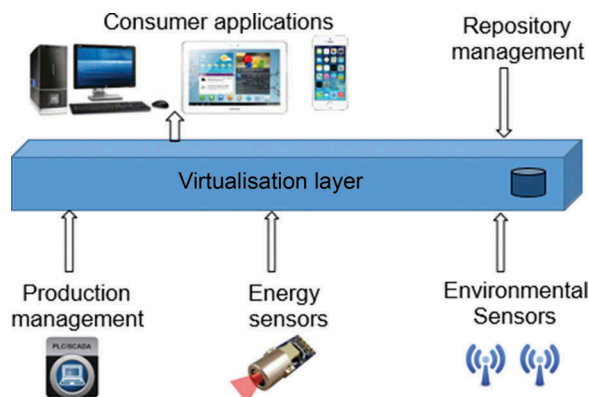


Figure 2. Consumers and information sources.

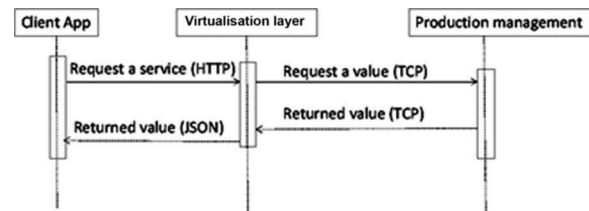


Figure 3. Getting values from a production management information system.

**8. Prototype implementation details**

The IT platform that has been used for services provisioning is GlassFish 2, a Java EE application server reference



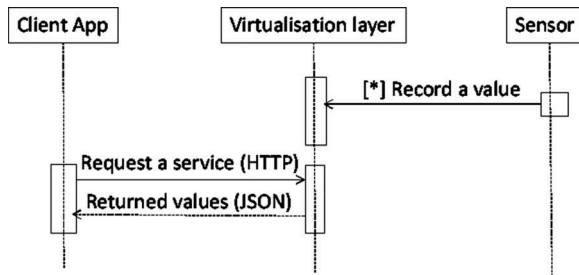


Figure 4. Sensor outputs accessible to client applications.

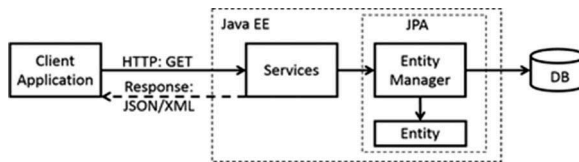


Figure 5. Java EE entity manager.

implementation. The database to which services connect is MySQL 3. The services are directly related to entities in the database. For instance, if there is a database entity for an enterprise supplying sensors, the entity may require services for adding a new enterprise, updating information of an existing one, for deleting an existing one, and for retrieving the information of a specific one, or even retrieving all the information of all companies.

The tight relationship between the database and the services is supported by a Java Entity Manager, which is provided by Java EE, in its Java Persistence API (JPA). The Java Entity Manager maps the database entity to a Java class, and also serves as proxy for executing the database operations. Queries to execute over the database are managed by an instance of the Entity Manager and – in case there are results – the results are automatically translated into class instances. Figure 5 shows how Java Entity Manager mediates access to the database and translates results into Entity classes. This technology is used to retrieve sensors' recordings from the database.

As mentioned before, services are grouped per entity. The code is very simple, as it just consists in a Java class with methods. Methods that will be exposed as services have 'annotations' specifying it. Every method could be a service, with their access path determined by the entity they belong to (Java 2013). The annotations specify the service performing the operation:

- POST – For inserting new data.
- GET – For retrieving data.
- PUT – For updating existing data.
- DELETE – For deleting data.

The format of execution path clients that should be specified to consume a service is: <http://root/FactoryEcomationServices/webresources/entities.company/count>. The base path defines the application and the package containing the entities, the path allows the server to find the place of entities it handles. Entities are annotated as stateless beans, and every method annotated with the operation it is associated to (GET, PUT, POST, and DELETE). In the previous path example, the specific entity is (company) and the service is (count).

The service path is not required if there is only one service for an entity. The service path is useful when there is more than one service from an entity. Even if the specific service is not specified in the URL, the request header already has this information, which will allow the identification of the service. There are also annotations for specifying the expected format of consumed and produced data. Data could be represented in JSON, XML or both. For services that could produce both formats, the client needs to specify which one it expects to receive in return.

## 9. Prototype testing results

This section describes the implementation of the ICT-infrastructure in which the SOA layer is a main component. The description is presented in a bottom-up way, by first derailing how the sensors' readings are managed, the services implemented and then how that information could be consumed by high-level applications.

Data readings from sensors are recorded in a database server to which the SOA layer has direct access. The database also keeps information about sensors details and their manufacturers, among other data. Storing sensors outputs is very important because knowing the last output is not enough; for pollutants it is necessary to gather information about pollutant trend along a time period, normally for the last hours and sometimes for the last few days. The SOA layer implements services for both possibilities.

The sensors are wireless, thus facilitating their installation at different places in the shop-floor. Every sensor is able to provide information about very basic characteristics, such as an ID, its type and reading range (minimum and maximum values it can read). At first deployment, a sensor provides such information to the sensor layer, so that information is recorded in the database server. By analysing recordings and data about sensors, the SOA layer can identify every new sensor, thus additional information about the sensor could be defined in the database, such as its location, manufacturer, etc. This functionality is similar to the concept of zeroconf (Cirani et al. 2014). However, sensors have a rather limited memory, thus not every aspect of a sensor could be specified in their internal memory.

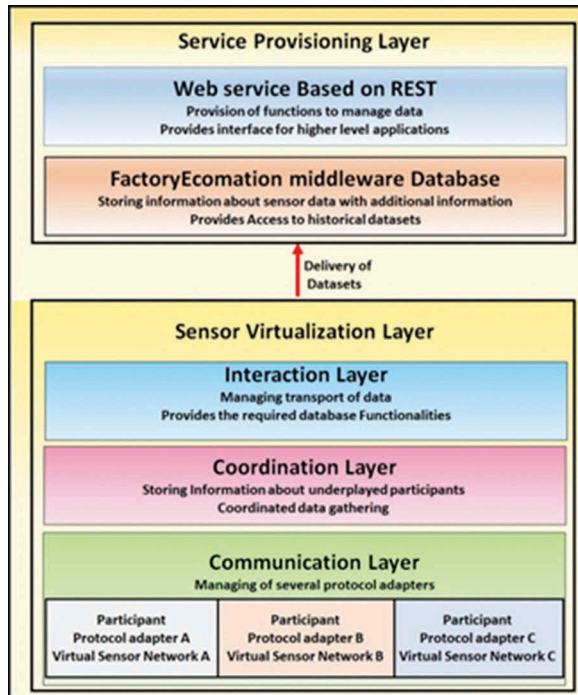


Figure 6. Virtual counterpart architecture.

The representation of the whole factory data for distributed applications is divided into two layers, the sensor virtualisation layer and the service provisioning layer. Together, these layers constitute the virtual counterpart architecture, depicted in Figure 6, which shows the structure of these two layers.

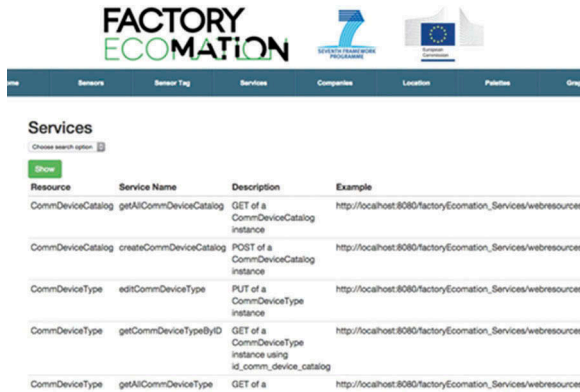
The Sensor Virtualisation layer represents the lower-level of the virtual counterpart. It gathers data from the sensors and delivers it to the database in the service provisioning layer. The Interaction sub-layer contains the set of functions and methods that enable and manage data delivery towards the service provisioning layer. The Coordination sub-layer manages communication with the physical sensors and requests access to the Communication layer. The latter coordinates the data flow from individual virtual sensor networks and stores the attributes of the whole communication structure according to the participating networks. The Communication sub-layer is divided into different parallel aligned regions where each of these areas is one participant in the total communication structure. The participants are introduced in order to represent groups of sensors in virtual form. The necessary parameters are stored inside the participants to maintain the communication to the directly associated sensor network. Data are exchanged via a defined communication protocol within the sensor network; thus, the participants of the Communication layer differ in the implementation of the communication protocol and in the characteristics of the parameters, which are stored in the participants themselves.

Figure 7. Sensor values management.

The Service Provisioning layer represents the high level of the virtual counterpart. It is responsible for storing sensors information in the database. All the information collected in this layer can be used by high-level applications. The Web Service based on RESTful sub-layer defines the SOA layer presented in this paper. Figure 7 shows one screen of a high-level application that connects to the database and manages sensors specification data.

As was mentioned before, one important requirement is simplifying the deployment of the ITC-infrastructure within different manufacturing enterprises, in which different consumer applications will be required. In order to achieve this, it would be necessary to find an easy way to locate available services. Currently, the amount of implemented services is 230, and this number would increase as new services are required by consumer applications. To solve this requirement, a services registration management application was developed, thus developers of new consumer applications will be able to search for services using different queries. Users could search for services by resource name, service name or description, among other search criteria. The registry application thus provides information about how a specific service could be consumed (e.g. called), the required input parameters and the result format, along with the specific URL that should be used, and an example of how to issue a call. Once the services descriptions are registered, users could search for resource name, service name or description, as is shown in Figure 8.

One issue found along the development was that users from different manufacturing enterprises use different names for concepts involved. A unification of concepts was necessary. To solve this, an ontology was developed after finding that there was not a consistent terminology in already available ontologies. Figure 9 shows the ontology structure. The ontology was used for the database names and as a dictionary to keep a consistent meaning in the information elements and what they represent.



The screenshot shows the 'Factory Ecomation' web application interface. At the top, there are logos for 'FACTORY ECOMATION', 'LEADER IN INNOVATION', and the European Union flag. Below the navigation bar, the 'Services' section is active, displaying a table of available services. The table has columns for Resource, Service Name, Description, and Example. The services listed include 'getAllCommDeviceCatalog', 'createCommDeviceCatalog', 'editCommDeviceType', 'getCommDeviceTypeByID', and 'getAllCommDeviceType'.

Resource	Service Name	Description	Example
CommDeviceCatalog	getAllCommDeviceCatalog	GET of a CommDeviceCatalog instance	http://localhost:8080/factoryEcomation_Services/webresources
CommDeviceCatalog	createCommDeviceCatalog	POST of a CommDeviceCatalog instance	http://localhost:8080/factoryEcomation_Services/webresources
CommDeviceType	editCommDeviceType	PUT of a CommDeviceType instance	http://localhost:8080/factoryEcomation_Services/webresources
CommDeviceType	getCommDeviceTypeByID	GET of a CommDeviceType instance using id_comm_device_catalog	http://localhost:8080/factoryEcomation_Services/webresources
CommDeviceType	getAllCommDeviceType	GET of a	http://localhost:8080/factoryEcomation_Services/webresources

Figure 8. Screen to search for details of available services.

- **Company**
  - **Sensor description**
    - Location
    - Description
    - Range of readings (min, max)
    - Precision of recordings
  - **Sensor data**
    - Sensor ID
    - Timestamp
    - Value
  - **Sensor archive**
    - Sensor ID
    - Timestamp
    - Value
- **Service description**
  - Service name
  - Description
  - URI
  - Example

Figure 9. Ontology structure.

Finally, in order to have a prototype to demonstrate the SOA layer functionality, a Web application was developed to show plots of the different information sources. The context model of the ICT-infrastructure is depicted in Figure 10. This model represents the dependencies among data and their users (Manager) or consumer applications; it also shows that 'Data recorder' application directly stores data from sensors, without the participation of a service, but users access the data through services. A layered perspective of the ICT-infrastructure was presented in Figure 2, showing the role of the virtualisation layer as an integrator of information sources of manufacturing operations including production, energy and pollutant agents.

As more data about sensors and services are registered, the deployment on new sites is simplified by not requiring

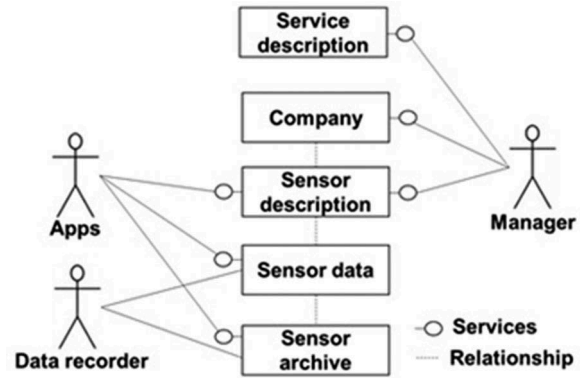


Figure 10. Context model of information and their users.

a complete description of potential sensors or services that may be used.

Some lessons learned from the implementation of the virtualisation layer and prototype applications are:

- A RESTful implementation of Web services for integrating different data sources indeed hides low-level details for consumer applications.
- The implementation of service documentation (e.g. a dictionary of available services) is of great help to provide information about available services, thus helping their consumption by new application clients.
- The development of an ontology is very useful in standardising the terminology and for defining names when developing a database.

## 10. Conclusions & future work

Manufacturing enterprises contribute significantly to both gross development product and labour in a significant way. However, it is very important that their operation does not jeopardise the geographic sectors in which they operate, by being one of the main sources of pollution being expelled to the environment.

This paper described a SOA-based infrastructure to integrate information sources about manufacturing operations, energy consumed and pollution control. All these three factors should be considered for the future eco-efficiency of manufacturing enterprises.

The diversity of manufacturing enterprises requires flexibility in simplifying the deployment of an ICT-infrastructure to integrate the diverse data sources relevant to eco-efficiency operation. The paper described how an SOA ICT-infrastructure can be complemented to simplify its deployment.

The paper describes the implementation of a services-based software layer to abstract the way information is

obtained from different data sources and consumed by application clients. A plotting application consumes sensors records at a medium rate; an undergoing emulator of a manufacturing factory plant will help the authors to evaluate eco-efficiency performance at higher rates for a hundred sensors. From the results obtained through the prototype applications, it is clear that the software layer implementation can satisfy the requirements for future application clients.

Another requirement that needs evaluation is the deployment of the ICT-infrastructure in different manufacturing enterprises.

### Disclosure statement

No potential conflict of interest was reported by the authors.

### Funding

The research presented in this document is a contribution for the Factory ECO-MATION (Factory ECO-friendly and energy efficient technologies and adaptive autoMATION solutions) project funded by the European Commission, FP7 FoF.NMP.2012-1, under the Grant Agreement No [314805].

### Note

1. Eco-efficiency – strategy that aims at minimising ecological damage while maximising efficiency of the production processes.

### References

- Aihkisalo, T., and T. Paaso. 2012. "Latencies of Service Invocation and Processing of the REST and SOAP Web Service Interfaces." In *IEEE 8th World Congress on Services*, 100–107. Honolulu, HI: IEEE.
- Amrina, E., and S. M. Yusof. 2011. "Key Performance Indicators for Sustainable Manufacturing Evaluation in Automotive Companies." In *IEEE International Conference on Industrial Engineering and Engineering Management*, edited by E. Qi, J. Shen, and R. Dou, 1093–1097. Singapore: IEEE.
- Bi, Z., L. Xu, and C. Wang. 2014. "Internet of Things for Enterprise Systems of Modern Manufacturing." *IEEE Transactions on Industrial Informatics* 10 (2): 1537–1546. doi:10.1109/TII.2014.2300338.
- Burke, B. 2009. *Restful Java with Jax-Rs*. Boston, MA: O'Reilly Media, Inc.
- Cirani, S., L. Davoli, G. Ferrari, R. Leone, P. Medagliani, M. Picone, and L. Veltri. 2014. "A Scalable and Self-configuring Architecture for Service Discovery in the Internet of Things." *IEEE Internet of Things Journal* 1 (5): 508–521. doi:10.1109/JIOT.2014.2358296.
- Davis, A., and D. Zhang. 2002. "A Comparative Study of DCOM and SOAP." In *4th International Symposium on Multimedia Software Engineering*, edited by B. Werner, 48–55. Newport Beach, CA: IEEE.
- Davis, C. 2012. "What If the Web Were Not Restful?" In *3rd International Workshop on RESTful Design*, edited by C. Pautasso, E. Wilde, and A. Marinos, 3–10. Lyon: ACM Press.
- de Oliveira, R. R., V. Sanchez, R. Vinicius, J. C. Estrella, R. Pontin De Mattos Fortes, and V. Brusamolín. 2013. "Comparative Evaluation of the Maintainability of RESTful and SOAP-WSDL Web services." In *7th International Symposium on Maintenance and Evolution of Service-oriented and Cloud-based Systems*, edited by A. Serebrenik, 40–49. Eindhoven: IEEE.
- Feinberg, R. 1950. "A Review of Transductor Principles and Applications." *IEEE-Part II: Power Engineering* 97 (59): 628–644.
- Fielding, R. T. 2000. "Architectural Styles and the Design of Network-based Software Architectures." Doctoral Diss., University of California, Irvine.
- Hoschek, W., and G. McCance. 2001. "Grid-enabled Relational Database Middle-ware." *Global Grid Forum* 3: 7–10.
- Java. 2013. "The Java EE 6 Tutorial" <http://docs.oracle.com/javase/6/tutorial/doc/gilik.html>
- Kim, H., S. Jeong, H. Yoe, N. Kim, S. Lee, and Y. Wen-zheng et al. 2014. "Design and Implementation of ICT-based System for Information Management of Livestock Farm." *International Journal of Smart Home* 8 (2): 1–6. doi:10.14257/ijsh.2014.8.2.01.
- Koho, M., S. Torvinen, and A. T. Romiguer. 2011. "Objectives, Enablers and Challenges of Sustainable Development and Sustainable Manufacturing: Views and Opinions of Spanish Companies." In *IEEE International Symposium on Assembly and Manufacturing*, edited by M. Lanz, 1–6. Tampere: IEEE.
- Linthicum, D. S. 2003. *Next Generation Application Integration: From Simple Information to Web services*. Sydney: Addison-Wesley Longman.
- Liskin, O., L. Singer, and K. Schneider. 2012. "Welcome to the Real World: A Notation for Modeling REST Services." *IEEE Internet Computing* 16 (4): 36–44. doi:10.1109/MIC.2012.59.
- Lomotey, R. K., and R. Deters. 2013. "Reliable Consumption of Web Services in a Mobile-cloud Ecosystem using REST." In *7th IEEE International Symposium on Service Oriented System Engineering*, edited by Y. Zhang and M. Younas, 13–24. San Francisco Bay, CA: IEEE.
- Mulligan, G., and D. Gracani. 2009. "A Comparison of SOAP and REST Implementations of a Service based Interaction Independence Middleware Framework" In *Simulation Conference*, edited by A. Dunkin, R. G. Ingalls, E. Yücesan, M. D. Rossetti, R. Hill, and B. Johansson, 1423–1432. Austin, TX: IEEE.
- Murer, C., and C. Szyperski. 2002. *Component Software-beyond Object-oriented Programming*. 2nd ed. Amsterdam: Addison-Wesley Longman.
- Pautasso, C., O. Zimmermann, and F. Leymann. 2008. "RESTful Web services vs Big' Web Services: Making the Right Architectural Decision." In *17th International Conference on World Wide Web*, 805–814. Beijing: ACM Press.
- Programme, U. N. E. 2014. "UNEP year book 2014: Emerging Issues in our Global Environment." <http://www.unep.org/yearbook/2014/>
- Qi, M. 2001. "Impacts of EDI on the Supplier." In *International Conference on Management of Engineering and Technology*, 50–59. Portland, OR: IEEE.
- Ren, L., L. Zhang, and F. Tao. 2010. "A Virtual SOA Model for Network Manufacturing Integration." In *International Conference on Mechanic Automation and Control Engineering*, edited by S. Qin, 3570–3573. Wuhan: IEEE.

- Rodríguez-Martínez, M., and N. Roussopoulos. 2000. "MOCHA: A Self-extensible Database Middleware System for Distributed Data Sources." *ACM Sigmod Record* 29: 213–224. doi:10.1145/335191.
- Shao, G., F. Riddick, J. Y. Lee, D. B. Kim, Y. Lee, and M. Campanelli. 2012. "A Framework for Interoperable Sustainable Manufacturing Process Analysis Applications Development." In *Simulation Conference*, edited by O. Rose and M. Adelinde, 1–11. Berlin: IEEE.
- Xie, P., and X. Gui. 2012. "Distributed Software Architecture of Manufacturing Integrated Service Platform based on SOA." In *International Conference on Systems and Informatics*, edited by X. Tong, 134–139. Yantai: IEEE.
- Xu, W., B. Yao, V. Fang, W. Xu, Q. Liu, and Z. Zhou. 2014. "Service-oriented Sustainable Manufacturing: Framework and Methodologies." In *International Conference on Innovative Design and Manufacturing*, edited by Y. Zeng, Y. Chen, and S. Achiche, 305–310. Concordia: IEEE.
- ZeroC. 2003. "Welcome to zeroc, the home of ICE." <http://www.zeroc.com/>
- Zhang, Y., H. Wang, S. Zhu, Y. Zhu, and P. Zhang. 2011. "Study on the Economic Development of the Industry Impact of Electricity and Heat Production and Supply Industry Pollution Abatement." In *International Conference on Electrical and Control Engineering*, edited by D. Fangmin, pp. 6131–6134. Yichang: IEEE.

Copyright of International Journal of Computer Integrated Manufacturing is the property of Taylor & Francis Ltd and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.