Position Papers of the Federated Conference on
Computer Science and Information Systems pp. 159–164

DOI: 10.15439/2015F37
ACSIS, Vol. 6

Reconfigurable FPGA-based embedded Web services as distributed computational nodes

Robert Brzoza-Woch *, Piotr Nawrocki †
*AGH University of Science and Technology,
al. A. Mickiewicza 30, 30-059 Krakow, Poland
e-mail:rabw@agh.edu.pl
†e-mail:piotr.nawrocki@agh.edu.pl

***Abstract*—In this article we propose a concept for an experimental class of control devices that use both a microcontroller unit (MCU) and a field-programmable gate array (FPGA) circuit. These devices can provide the functionality of full-featured Web services that are compliant with the Service-Oriented Architecture (SOA) paradigm. Despite the fact that FPGA circuits are more expensive than consumer-grade MCUs, they potentially offer much more computational power. In scenarios in which FPGA computational power is required on demand and for short periods only, a large part of such resources might, however, remain unused or disabled. Thus we propose a system architecture and software infrastructure that simplify the utilization of temporarily unused resources for performing various tasks that can be offered as Web services on a commercial basis.**

I. Introduction

FPGA-based hardware Web services have already been implemented and described (refer to [1] and [2]). Their embedded nature allows developers to easily adapt those services to actively interact with their environment, e.g. to acquire real-world measurement data or control various actuators. Such entities can be called *environment-aware* Web services in contrast to classical Web services that work on remote physical or virtual machines. Despite the fact that environment-aware Web services may be implemented using much less expensive MCUs and sequential code, programmable hardware may perform better where very intensive computational tasks are involved.

In our solution we propose that environment-aware Web services can be reconfigured in order to exploit the potential of their temporarily unused logic resources. At times of lower utilization they can be reconfigured to offer their spare resources as additional data-processing Web services. Whenever a more intensive processing task is to be performed, their resources can be employed back to provide the device's original functionality. This idea can also be applied to regular devices that offer no Web service compliance. In the latter case, however, we would lose some useful features such as interoperability or the ability to utilize the management software tools already available, etc.

There are common scenarios where considerable computational power is required only on demand for a short period. For example, in an industrial process temperature measurement results are collected for many hours to finally update a local numeric prediction module. During the time of collection, the acquisition device may be idle or in sleep mode or, if it uses FPGA, a large part of its logic may be disabled. By using the dynamic reconfiguration technique for FPGA, we can change the device's configuration depending on momentary needs, e.g. to adapt to changing environment conditions or the current context.

FPGA-based environmental-aware Web services may also be employed to perform control tasks in smart home automation systems. The less demanding the control task, the more resources can be assigned to perform "idle" computations. For example, an intelligent water heater may periodically compute predicted hot water usage and perform computations for a commercial Web service during its "spare" time.

In this paper we describe the concept for a system that uses FPGA-based Web services to perform such dual operation. In Section II we present the current state of the art in related fields. Then in Section III we describe the architecture of sample FPGA-based Web services and discuss the architecture of the entire system. In Section IV we introduce the service management and integration mechanism, which is responsible for providing the essential functionality of the system and ensuring its security. Finally, in Section V we conclude our work and discuss the planned evolution of our solution.

II. Related work

Web service implementations using FPGAs can be found, however such publications are relatively rare. In [3], the authors propose a Web server architecture that is implemented in FPGA. The evaluation of the system confirms that hardware-favored architecture brings higher throughput, lower power consumption and the full functionality of a stand-alone Web service. Such good results are achieved thanks to the execution of Web services directly on the FPGA without using an additional operating system. The authors conclude that by utilizing reconfigurable hardware (FPGA) in the area of cloud computing it is possible to improve performance and optimize operating costs.

Important research in this field is described in [4]. The authors present a reconfigurable architecture for Web service implementation. The important features of the system presented are:

- high overall performance because of the very low response time and potentially high processing power;
- a reconfiguration ability which allows the system to be updated to meet new requirements.

Both advantages mentioned result from the use of the FPGA technology. The platform supports the SOAP protocol and is able to auto-register into a UDDI server. Even more interestingly, the platform presented works without any embedded microcontroller (such as NIOS-II) yet it is partially implemented using the Handel-C language. The overall performance results are good because the platform has a lower response time (a minimum of 0.5 ms) than a PC running the same service (a minimum of 2 ms). A disadvantage of the solution described is its very simple functionality—in the configuration presented, it only provides a Wake on LAN service for computers within a local area network.

Another FPGA-based Web service implementation is described by C.E. Chang in [5]. It is a RESTful Web service designed to perform simple control tasks for home appliances. The service functionality is rather minimalistic and its implementation has less features than the system described in [4], e.g. it supports static IP assignment only (no DHCP), the TCP/IP buffer size is limited to just a single packet or 576 bytes, and only one TCP connection can be accepted at a time.

The possibility of using reconfigurable hardware and Web Service technology within the framework of the concept of the Internet of Things (IoT) is described in [6]. The authors of that paper discuss the prospects of reconfigurable hardware solutions in the area of enterprise applications. They present requirements for reconfigurable computing solutions and argue that this type of platform can assist the performance of business processes. They also estimate that reconfigurable computing platforms will play a key role in connecting two worlds: the “Internet of Services” and the “Internet of Things”.

In [7], the authors note the need for connecting the concept of IoT with service-oriented methodology. They suggest the use of RESTful Web services due to their popularity and lightweight nature. In order to utilize the RESTful concept in the IoT domain, the article proposes an architecture composed of six levels. The authors also demonstrate how to invoke RESTful Web services from the IoT and publish a Business Process Execution Language (BPEL) process as a RESTful Web service.

Important areas that utilize the IoT and Web service concepts are the Smart Home and Smart Building. In order to provide IoT services in Smart Home/Smart Building environments, in [8] the authors propose a Web-of-Objects platform in the IoT service environment. This platform has been designed in order to create user-centered IoT services. In addition, complex services can be developed by combining elements of existing Web services.

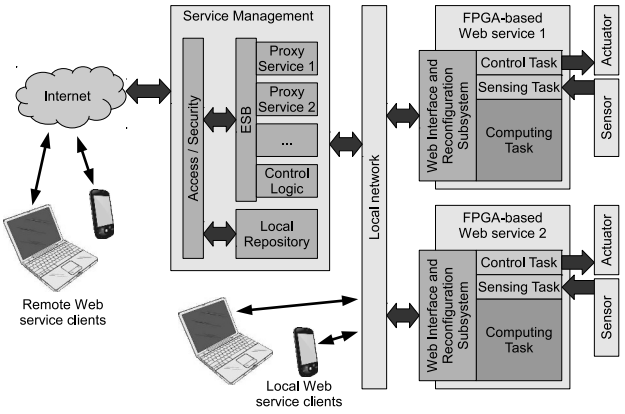

Fig. 1. General concept of networked reconfigurable FPGA-based Web services.

III. Hardware infrastructure and system architecture

As stated in Section I, SOA-compliant Web services may be implemented not only on classical server farms, but also on embedded devices, obviously including FPGA-based ones. During the course of our research we developed several FPGA-based Web services on various hardware platforms. One of the most versatile hardware platforms we developed was described in detail in our previous publications: [1] and [2]. Each platform was equipped with a high-end Stratix-II FPGA chip and multiple blocks of synchronous dynamic random access memory (SDRAM) and was able to support built-in as well as external sensors and actuators. In this article we describe how we experimented with the deployment of those platforms within dynamically managed and reconfigurable distributed systems.

A. System architecture

The architecture of the distributed control computing system is shown in Fig. 1. In that scenario, FPGA-based Web services perform several tasks. First, they provide part of the Web service interface and perform their default control and sensing tasks that require only a fractional amount of their logic resources while their spare resources are assigned to perform a *Computing Task*. We assume that the computing task is required for the process being supervised only during certain periods, for example after a sufficient amount of data has been collected – this is quite a common scenario in computer systems. During the service’s lower activity periods, its resources can be assigned to perform a completely independent task thanks to the massive parallel capabilities of the FPGA technology.

FPGA-based Web services can be conveniently interconnected within a local area network or may operate in a virtual private network (a VPN, which is logically equivalent to local area network operation). At this basic level, data transmission security can be ensured either using the Wi-Fi Protected

Access (WPA) technology for wireless entities or by using wired connections (it is fair to assume that an attack on a building's backbone network is equally invasive as tampering with physical devices). Depending on security requirements, we may or may not allow trusted clients to have unrestricted access to FPGA-based Web services in the local area network. Access restrictions can be introduced using various techniques, even very simple ones: e.g. by means of Media Access Control (MAC) address filtering.

More sophisticated features are available when using the *Service Management* subsystem shown in Fig. 1. The management subsystem runs on a stand-alone computer or on server infrastructure and it provides *Proxy Services* for each physical or logical Web service on the network. Proxy Services facilitate access to local embedded services by increasing the number of simultaneous client connections and introducing the Enterprise Service Bus (ESB). ESB significantly increases the solution's scalability and accessibility from mobile devices. Separate services are connected to the ESB to provide control logic and access to the local service repository. The *Control Logic* service on ESB is the core of our concept. It decides whether an FPGA-based Web service should be reconfigured and which of these services can be offered as spare computational resources to the outside world. The logic may also advertise available resources to service brokers. As computational services may not be always available on the programmable hardware (during idle periods only), the control logic should be able to move the execution of tasks from FPGA-based services to its internal software and back to the FPGA after hardware resources have been released again. At lower level, this functionality can be easily implemented on various existing FPGA platforms with additional reconfiguration hardware. Another challenge concerns the algorithms that support decision whether the functionality should be moved to or from FPGA to the control logic. Those algorithms may vary from a single busy-idle state detection to more advanced solutions which also consider potential time and energy costs of the task moving operation. That functionality, however, is still under development and is to be introduced in future versions of the system.

B. FPGA-based Web service architecture

In order to cooperate within the system presented, each FPGA-based embedded Web service should follow some particular architectural recommendations. The most trivial issue with the FPGAs' remote reconfiguration is that most common FPGAs completely lose their previous functionality during the process of reprogramming their internal memory. This is the reason why all FPGA-based Web services should be implemented as *hybrid* devices. A simple way to ensure uninterrupted operation in this case is to implement part of the communication interface and reconfiguration logic on a different chip. In the most trivial case, the reconfiguration logic can be implemented on a popular inexpensive microcontroller unit (MCU) with network communication capabilities as shown in Fig. 2.

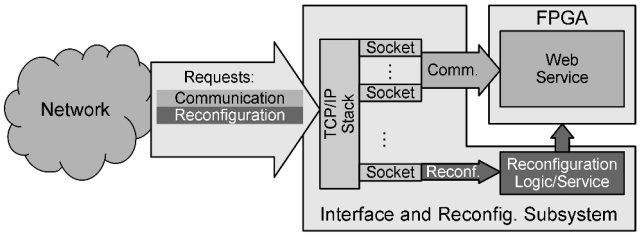

Fig. 2. The idea of multiplexing network hardware between the FPGA-based Web service and the reconfiguration subsystem.

In practice, a relatively simple and effective solution was to introduce multiplexing at the network socket level by implementing communication functionality up to the transport layer of the Open System Interconnection (OSI) model. Depending on the available resources of the reconfiguration subsystem, performance constraints may occur. In practical implementations it was not the case, for two reasons. First, FPGA-based Web services tend to process data locally whenever possible and transmit processing results only. For example, a smart camera, which we have previously implemented (refer to [2]), runs a motion detection or object classification service. Instead of transmitting an entire video frame, the camera sends mainly coordinates for which it detected motion or the identification number of the object recognized. Secondly, each hardware service is represented by its "mirror" proxy service in the Management subsystem. This allows the hardware service to be exposed by a machine that has better networking capabilities than a typical embedded system. Reconfiguration data also do not have to be transmitted for each functionality, but can be cached locally on each node instead as described further in this section.

The reconfiguration logic of FPGA-based Web services can be exposed either as a constant method available for each service or else can use an application-specific protocol. Both these options have their advantages. Exposing reconfiguration capability as a service makes for a very clean system design and ease of integration as a proxy in the Management block. However, using a simple custom protocol may offer better performance because it does not involve the high-level protocol overhead that is unavoidable when using e.g. SOAP. A simple but efficient enhancement of the reconfiguration subsystem is the introduction of mass storage capability, e.g. in a form of a popular secure digital (SD) card or another form of non-volatile memory. This allows the system designer to cache the most common configuration data locally for each hardware Web service. In our solutions we used SD and SD High Capacity (SDHC) cards as well as DataFlash memory manufactured by Atmel.

Network communication can be implemented in several ways. The obvious choice for wired nodes is to use an Ethernet connection for the easiest possible interoperability with the network infrastructure already available. In our solutions, we

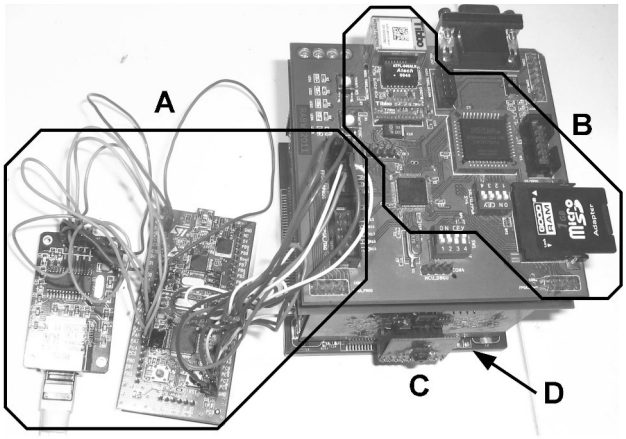

Fig. 3. View of the hardware part of sample implementations: the newly developed inexpensive remote reconfiguration and communication subsystem (RCS) based on the ENC28J60 and a mid-end STM32 MCU (A), the previously developed RCS based on the EM1206 network module, an STM32 MCU and CPLD, the embedded image sensor (C), and the Stratix-II FPGA module which provides Web service functionality (D).

initially tested in practice the operation of EM1206 network modules by Tibbo. These offer very simple programming capabilities thanks to their high autonomy and can be connected to any device (either an FPGA or a microcontroller) that supports asynchronous serial communication. Another simple solution is to use the ENC28J60 chip, which provides physical layer (PHY) capabilities and is equipped with a synchronous serial interface. Upper layers should then be implemented in microcontroller software. More demanding applications would require the utilization of a PHY chip with a fast Media Independent Interface (MII). Wireless network access can be provided using a Wi-Fi chip or module, e.g. ESP8266. It is a very inexpensive and easy-to-use module which became very popular recently. The ESP8266 makes it possible to connect virtually any embedded device to a Wi-Fi network and, importantly, has basic connection security measures already implemented. In our research we developed multiple FPGA-based Web services using EM1206 (wired) and ESP8266 (wireless) modules. Now we continue to further develop our solutions using other alternatives, mainly revolving round medium-power microcontroller (such as Cortex-M3) with external PHY chip. Fig. 3 shows a sample inexpensive hardware with ENC28J60 PHY and STM32F100 board that we have used in our development.

IV. Service Management Subsystem Details

The considerable computing capabilities of FPGA-based Web services make it worthwhile to enable them to interact and integrate with other systems. This type of solution, integrating independent but compatible services, lies at the foundation of the SOA concept and the Web service technology is one of the components of this concept. SOA may take advantage of service orchestration, which defines the model of cooperation between services. Within the framework of orchestration, there is a single parent process that manages the interoperability between services. The implementation of such a process, which allows for the provision of management and supervision services, is achieved via the Enterprise Service Bus (ESB), which specifies an intermediate layer that enables the integration of services. In order to enhance the processes of composing and configuring FPGA-based Web services, Proxy Service is created depending on current needs. This mechanism allows for the provision of FPGA-based Web service functionality using the ESB. The main advantages of this solution are standardization, scalability, reliability and manageability. The Proxy Service, which is implemented in Java, exposes the functionality of an FPGA-based Web service in an ESB container and allows it to better integrate with other services. Most Java implementations of ESB use the OSGi platform that allows, inter alia, the re-use of components, easy deployment and the ability to dynamically update components.

TABLE I
RESTful and WS* service comparison.

RESTful	WS-* (SOAP)
Lightweight architectural style	"Heavy-weight" XML standard
Description of the service in WSDL 2.0	Description of the service in WSDL
Format agnostic (XML, JSON, HTML, etc.)	Requests and responses are well structured (SOAP)
Problems with reliable messaging and security	Security mechanism possible (WS-Security)

In order to implement FPGA-based Web services and ensure their cooperation with the Proxy Service mechanism we could use two approaches: WS-* using SOAP and Representational State Transfer (RESTful) Web services built on the basis of HTTP. These are briefly compared in Table I. The first method describes the functionality of the service using the Web Service Description Language (WSDL), and communication with the service is implemented using the SOAP and HTTP protocols. The WS-* concept was originally developed for interoperability between enterprise applications. A lighter version of the WS-* was developed, named Devices Profile for Web Services (DPWS), in which services are directly associated with the equipment. The main DWPS area of application is home and industrial automation systems ([9]). In the second approach, RESTful services are identified by a Uniform Resource Identifier (URI) and the HTTP protocol is used to access the resources thus defined. The RESTful approach can be used to install services on smart devices. Some studies such as [10] suggest that for smart devices used in the development of IoT applications, a more suitable method for the provision of services is RESTful. On the other hand, the WS-* concept is more appropriate for applications requiring an adequate level of security and QoS. Therefore the authors of this article decided to choose the WS-* approach due to the potential application of the solution developed in systems that require reliable and secure data delivery.

After further analysis, we decided to use WS * and OSGi containers as our ESB implementations. Integration and ser-

vice management have been achieved through the exposition of FPGA-based hardware Web services using the Proxy Service—a service engine that is implemented in Java. The Proxy Service has been deployed within the ESB (OSGi container). The functionality of the service can also be exposed directly in a local area network as a Web service reachable using SOAP communication standards. In both cases, the interface can be specified in WSDL.

In our solution, each of the FPGA-based hardware Web services is described by an appropriate WSDL file. For each such service, it is possible to generate one corresponding Proxy Service with methods identical to those of the original service. All operations on a Proxy Service are delegated through standard WS requests (SOAP) to the server appropriate for the target device. Recently we have also developed experimental REST-based implementations of the embedded Web services because of the REST's simplicity and smaller communication overhead.

The automated generation process of a Proxy Service requires several steps:

1) generating a Proxy Service as an ESB adapter;
2) looking up a matching service reference in repositories;
3) downloading a WSDL file that describes the service's interface;
4) creating a hardware service interface in the Proxy.

Afterwards, the FPGA-based hardware Web service is available and ready to be used in enterprise class information systems.

The functionality of the FPGA-based hardware Web service is accessible through the Proxy Service mechanism thanks to mediation between the SOAP binding component and the OSGi Remote service binding component such as the R-OSGi (see [11]) or ECF Remote Services. The use of such ESB implementations as Fuse ESB 4.3, Apache ServiceMix 4.3.0 or Apache Felix (installed on mobile devices supporting the Android system) enables direct access to these services through the OSGi environment.

In addition to providing the functionality of the FPGA-based hardware Web service, each Proxy Service provides an interface that also allows for the remote reconfiguration of the FPGA device and provides completely new functionality. Details of the reconfiguration process are presented in Section III.

A very important issue in the integration and management of FPGA-based hardware Web services is the discovery process. All FPGA-based hardware Web services use network communication modules that provide a multi-socket TCP/IP stack. Some sockets have fixed roles and the rest are used for clients' connections. One socket is reserved for discovery purposes, enabling the FPGA-based hardware Web service currently running to be found within the local area network. There is a local (service) repository responsible for maintaining a directory of FPGA-based hardware Web services, detecting new services connected to the system and managing service configuration. The prototype service repository used in our solution utilizes the LDAP (Lightweight Directory Access Protocol) for FPGA-based hardware Web service inventory management purposes. Each device in the system is obliged to send beacon signals periodically. These beacon signals are sent as UDP datagrams to a multicast destination IP address and received by the service repository. Contents of each beacon message are generated in the internal logic of each individual FPGA-based hardware Web service and sent through the network communication module, which then transmits them using UDP.

An internal user who intends to interact with an FPGA-based hardware Web service browses the service repository in search of the service's details such as the invocation point and the signature of the operations supported. Upon finding them, the client application can connect to the hardware service directly. An external user can interact with an FPGA-based hardware Web service (find and run it) through the Internet and ESB.

A significant aspect of using FPGA-based Web services concerns the construction of a proactive system. Both protocols investigated (RESTful and WS*) have request/response characteristics that do not support the generation of asynchronous notifications to clients. The fact that the classic approach to Web Service technology has been selected implies that we cannot implement such a notification mechanism in our solutions. Notwithstanding this, in the concept developed we have the ability to send notifications on service status via the local (service) repository that is found under a well-known address, which acts as a proxy that performs notification operations. In the solution proposed, FPGA-based Web services automatically send registration information to the repository. The same method can be used to send information about service status to the repository. In this case, a separate notification mechanism can be implemented in the repository. Such a solution can be perceived as a step towards event-driven services.

During the design and implementation of our chosen solution, we also considered issues related to security. It was assumed that the local area network, which is not connected directly to the Internet, is safe. If secure communications in the local area network are required, SOAP and REST can be used over HTTPS. If wireless technology is used, security is provided by standard mechanisms such as Wi-Fi Protected Access (WPA / WPA2) as implmented in the ESP8266 modules that we use, Wireless Intrusion Prevention Systems (WIPS) or Wireless Intrusion Detection Systems (WIDS). User access from the Internet to FPGA-based hardware Web services is possible only through the ESB, where various user authorization and authentication mechanisms can be implemented through secure communication channels such as the HTTPS protocol.

V. Conclusions and future work

In this article, we present a concept for a better utilization of spare FPGA resources by employing them to perform independent computational tasks. We apply this approach to FPGA-based embedded and environment-aware Web services compliant with the SOA paradigm. Additional functional

modules have to be provided for each service and particular architectural guidelines have to be followed, which we present in this paper as a reference. We attempt to keep additional hardware costs as low as possible. Initially, we applied the concept presented to the previously developed FPGA hardware-software platform designed to run various Web services. Future development goals include 1) automatic service advertising (which is related to the issue of service repository [12]); and 2) developing or adapting available algorithms which would allow us to automatically move computations between FPGA-based Web services and the service management subsystem to ensure uninterrupted Web service operation.

ACKNOWLEDGMENT

The research presented in this paper was partially supported by the European Union in the scope of the European Regional Development Fund program no. POIG.01.03.01-00-008/08, the National Centre for Research and Development (NCBiR) under Grant No. PBS1/B9/18/2013 and by the Polish Ministry of Science and Higher Education under AGH University of Science and Technology Grant 11.11.230.124 (statutory project).

REFERENCES

[1] A. Ruta, R. Brzoza-Woch, and K. Zieliński, "On fast development of FPGA-based SOA services—machine vision case study," *Design Automation for Embedded Systems*, vol. 16, no. 1, pp. 45–69, 2012. doi: 10.1007/s10617-012-9084-z. [Online]. Available: http://dx.doi.org/10.1007/s10617-012-9084-z

[2] R. Brzoza-Woch, A. Ruta, and K. Zieliński, "Remotely reconfigurable hardware-software platform with web service interface for automated video surveillance," *Journal of Systems Architecture*, vol. 59, no. 7, pp. 376 – 388, 2013. doi: 10.1016/j.sysarc.2013.05.007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138376211300074X

[3] J. Yu, Y. Zhu, L. Xia, M. Qiu, Y. Fu, and G. Rong, "Grounding high efficiency cloud computing architecture: HW-SW co-design and implementation of a stand-alone web server on FPGA," in *Applications of Digital Information and Web Technologies (ICADIWT), 2011 Fourth International Conference on the*, Aug 2011. doi: 10.1109/ICADIWT.2011.6041412 pp. 124–129.

[4] S. Cuenca-Asensi, H. Ramos-Morillo, H. Lloren-Martinez, and F. Macia-Perez, "Reconfigurable architecture for embedding web services," in *Programmable Logic, 2008 4th Southern Conference on*, March 2008. doi: 10.1109/SPL.2008.4547742 pp. 119–124.

[5] C. Chang, F. Mohd-Yasin, and A. Mustapha, "An implementation of embedded RESTful Web services," in *Innovative Technologies in Intelligent Systems and Industrial Applications, 2009. CITISIA 2009*, July 2009. doi: 10.1109/CITISIA.2009.5224244 pp. 45–50.

[6] M. Middendorf and B. Scheuermann, "Perspectives of extending runtime reconfigurable computing to the enterprise application domain," in *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, July 2010. doi: 10.1109/INDIN.2010.5549416 pp. 266–273.

[7] L. Zhang, S. Yu, X. Ding, and X. Wang, "Research on IOT RESTful web service asynchronous composition based on BPEL," in *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2014 Sixth International Conference on*, vol. 1, Aug 2014. doi: 10.1109/IHMSC.2014.23 pp. 62–65.

[8] Y. Kim, S. Lee, Y. Jeon, I. Chong, and S. H. Lee, "Orchestration in distributed web-of-objects for creation of user-centered iot service capability," in *Ubiquitous and Future Networks (ICUFN), 2013 Fifth International Conference on*, July 2013. doi: 10.1109/ICUFN.2013.6614920. ISSN 2165-8528 pp. 750–755.

[9] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation," *Industrial Informatics, IEEE Transactions on*, vol. 1, no. 1, pp. 62–70, Feb 2005. doi: 10.1109/TII.2005.844419

[10] D. Guinard, I. Ion, and S. Mayer, "In search of an internet of things service architecture: REST or WS-*? a developers' perspective," in *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, A. Puiatti and T. Gu, Eds. Springer Berlin Heidelberg, 2012, vol. 104, pp. 326–337. ISBN 978-3-642-30972-4. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30973-1_32

[11] J. S. Rellermeyer, G. Alonso, and T. Roscoe, "R-OSGi: Distributed applications through software modularization," in *Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware*, ser. Middleware '07. New York, NY, USA: Springer-Verlag New York, Inc., 2007. doi: 10.1007/978-3-540-76778-7_1 pp. 1–20. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-76778-7_1

[12] P. Nawrocki and A. Mamla, "Distributed web service repository," *Computer Science*, vol. 16, no. 1, pp. 55–73, 2015. doi: 10.7494/csci.2015.16.1.55. [Online]. Available: http://journals.agh.edu.pl/csci/article/view/1205