# Building the Web of Things with WS-BPEL and Visual Tags

Web of Things using Service-oriented Architecture standards

Antonio Pintus, Davide Carboni, Andrea Piras
ICT-Information Society
CRS4
Pula, Sardinia, Italy
e-mail: pintux@crs4.it, dcarboni@crs4.it, piras@crs4.it

Alessandro Giordano
Dip. di Ingegneria Elettrica ed Elettronica
Università degli Studi di Cagliari
Cagliari, Sardinia, Italy
e-mail: alegiordy@tiscali.it

***Abstract*— The Web of things is an emerging scenario in which everyday objects are connected to the Internet and can answer to HTTP queries with structured data. This paper presents a system that allows users to build networks of everyday objects using visual tags as proximity technology. The system backend is based on Service-oriented Architecture languages and tools for the runtime composition of "things" establishing connections we call hyperpipes.**

***Keywords - Ubiquitous computing, Web services, Web of Things, SOA***

## I. INTRODUCTION

The Web of things is an emerging scenario in which every object is connected to a pervasive wireless/wired network and can answer to a HTTP query with structured data. Everyday surrounding objects like phones, domestic appliances, advertisement billboards, musical instruments become the nodes of the Web of things. Nevertheless, merely putting real objects into the network is nothing without a logic that creates a net value. One key is to compose objects together [10] and to put the orchestration in the hands of the final user. Simple mechanisms to connect "things" can foster a huge number of unpredictable applications. Towards these objectives, users, objects and networks are the ingredients to build a Web of things in which users become seamlessly the "programmers" [9]. The Web of things has the potential to become the next killer application and it must seamlessly emerge from existing Web infrastructure taking to the limits all the Web related technologies and providing new use cases that will improve the definition and the adoption of new standards and protocols.

The Web is basically built on two metaphors: the hypertext and the hyperlink. The former is just a digital reification of human language in its written form, while the second is a mechanism that non-digital forms of writing (like writing on paper) could not provide. The two metaphors are on the basis of the Web of pages while in a Service-oriented Architectures (SOA) not only pages are linked together but are also linked with information services. We want to extend the pages and services interlink from digital objects to real ones.

This paper, after an analysis of some related works and an introductive classification of "things" based on their capabilities, presents a "things" composition framework called hyperpipes. Finally, a prototype and conclusions along with indications for future work are provided.

## II. RELATED WORKS

This work is located in the main stream of ubiquitous computing, and more precisely in a subset of the "Internet of Things" based on Web protocols instead of ad-hoc and special purpose transport and application protocols.

We think that in the Web of things all kind of services (WS-* and REST [5]) provided by objects must be orchestrated together but for practical reasons in this work we sketch a design solely based on Web services Description Language (WSDL) and Simple Object Access Protocol (SOAP) Web services. The use of SOA Web standards for the Internet of things is not new: the SODA project [4] goes toward the definition of an architecture where devices are viewed as services in order to integrate a wide range of physical devices into distributed IT enterprise systems. A SOA approach for embedded networks is also persuaded by other projects, such as SIRENA [8] and SOCRADES [13]. Our work distinguishes from the others above because we experiment the direct generation of new process definitions according to user selection and pointing of real objects in the environment. We postulate that physical objects must expose in a formal specification the set of operations they can perform and the data they can exchange with a precise contract in a way that they can be composed and orchestrated using existing standard languages like the Web services Business Process Execution Language (WS-BPEL) [1]. This assumption makes the choice of SOAP and WSDL 1.1 of practical use for our actual implementation. In principle, the inclusion of RESTful services in orchestration is possible with the support of WSDL 2.0 but in practice this standard cannot be effectively adopted yet. In the meanwhile RESTful services could be proxied by ad-hoc SOAP services and orchestrated in WS-BPEL but in this work we do not address this issue.

The proximity of users to objects is another fundamental aspect that must be considered in pervasive computing. One of the peculiar points of our work is that process definitions for object pipelining are created by users on demand. In [11] the authors use Bluetooth as option for providing connectivity, and propose RFID technology to enhance the Bluetooth connection establishment procedure.

 ISBN: 978-1-61208-100-7 

Our approach to proximity is that after an object and in a given situation and with a given mood a person can have the idea to build something new. The subsequent action in our scenario is to build a connection. If we imagine the world as a giant sketch board we just want a way to draw a line from an object to another and build something useful as result. In [3] a similar interaction pattern is depicted but the system architecture and the data formats are described at a general level while in this work we focus on architectural aspects with formal specification and adoption of SOA standards. Simple but effective rules, applied to a multitude of objects tend to form a complex system [7]. In our scenario millions of real objects can simply be connected through hyperpipes with natural gestures in the real environment without sitting in front of a PC screen.

## III. ASSUMPTIONS ON OBJECTS CAPABILITIES

One of the main assumptions in a Web of things is that objects can communicate at HTTP level and above. This assumption is a weak one because technical solutions to achieve this result are already discussed and designed in literature [2], [12], [14] and some projects are ongoing. Thus, if the connectivity is lacking in the real world this is due to a lack of infrastructure and not to a lack of know-how.

Nevertheless, it is useful to sort "things" according to the level in the communication stack they can be connected:

- at the top of this sorting we have bare virtual goods and services like Web sites, e-mail boxes and 3D models, just to mention some. These objects can be easily wrapped and then referenced in a HTTP addressing space like resources (REST) or like services (WSDL).
- At a second level we find appliances with a complete HTTP stack like wireless printers or networked screens.
- In a upper intermediate level we find objects that are not equipped with a complete HTTP stack but can still communicate at a TCP/IP or UDP/IP level. For those objects is straightforward to build a HTTP wrapper.
- In a lower intermediate level we find objects that cannot communicate over IP networks, but still can communicate with different protocols like ZigBee, Bluetooth or X10. For those objects a proxy can be deployed to present these objects in the HTTP addressing space.
- Finally, there are bare physical objects. For those a digital counterpart must be built and published online. For example, a real book has a virtual counterpart like a Web page in a online bookstore.

Let us consider any object (physical or digital) like a process exposing a set of operations. We classify the operations according to their ability to produce data (sources), process data (processors), and consume data (sinks). This classification is useful to distinguish between sensors, actuators and processors. Operations are public, thus their names are globally known.

## IV. HYPERPIPE FRAMEWORK BASED ON SOA LANGUAGES

Considerable challenges are related to connecting a large set of information sources and sinks together. When only existing protocols and data formats are used, the communicating parties must be matched based on the descriptions describing their capabilities. To get a balance among the generality of purposes and the need to implement a system really able to work, we made some choices that drive the design of our work. First, we choose to adopt WSDL as the formalism to describe what an object is able to provide. In this way, an object can be considered as a SOAP Web service. Another issue is related to the type of communications between objects and the definition or the adoption of a suitable related protocol. In our design objects are allowed to exchange data without strict type checking (automatic type adaptation is a feature), and communication may be either synchronous or asynchronous. Both these interactions can be easily modeled and implemented using WSDL and adopting SOAP over HTTP protocol for messages exchange. Another type of logical connection to include in the design is multimedia streaming between objects. For instance, the user selects a MPEG camera source and a wall screen as sink. Embedding multimedia streaming in SOAP messages is not an efficient implementation, thus another protocol should be used instead. In concrete, the main assumption we have to make is that, in order to actively play a role in a pipe, an object must be able to connect to the network and to run a Web service stack. This general capability can be accomplished basically in two ways: the object itself is powerful enough to satisfy the previous requirements or it has to be connected and "driven" by a proxy computer, which satisfies the requirements. We choose WS-BPEL for concrete representation of pipes. WS-BPEL is an XML-based language born to define executable business processes as orchestration of Web services. WS-BPEL orchestrations expose a service interface described using WSDL: in this way, from the point of view of a client, WS-BPEL process is a Web service itself. Expressing pipes using WS-BPEL brings two main benefits to our vision: first, it is possible to associate a well-defined functional interface to each pipe, in our case modeling that in order to expose Video Cassette Recording (VCR)-like functionalities: start, pause and stop, which are the public available operations for a generic pipe control.

Three basic patterns of "in-Pipe" communication emerge:

*a) synchronous, on an object A is invoked an operation src, the result is adapted and then passed as argument to an operation snk of an object B;*

*b) asynchronous, the pipe registers itself as a listener for an event produced by an operation src on object A. When the event is fired, the data attached to the event is adapted and the sent to the sink.*

*c) streaming, an object B receives from an object A a stream of data (for instance video mpeg from a camera to a screen). Given that binary real time data encapsulation inside SOAP messages is not an efficient implementation, rather Real-time Transport Protocol (RTP) or equivalent*

 ISBN: 978-1-61208-100-7 

*real time media transmission protocols should be used instead, using the SOAP messaging only to initialize the session for handshaking.*

For the first two patterns, we created two different WS-BPEL document templates, which define all the required activities, message exchange and service orchestration for the execution of each of them in a WS-BPEL engine. In particular, pattern (a) is a typical Web service orchestration scenario with a subsequent invocation of services; pattern (b) basically is an orchestration in which the WS-BPEL document describes an asynchronous invocation of a service (the event producer) using a callback mechanism, which invocation triggers an event causing a delivering of the event to the other service (the event listener). The pattern (c) uses WS-BPEL only for protocol negotiation and handshaking between the two services, in this way, after these steps, the objects can instantiate streaming sessions in an independent way using the suitable chosen protocol. From a WS-BPEL point of view the pattern (c) is equivalent to pattern (a) but the data exchanged are Session Description Protocols (SDP) instances and the work of establishing a streaming is completely delegated to endpoints.

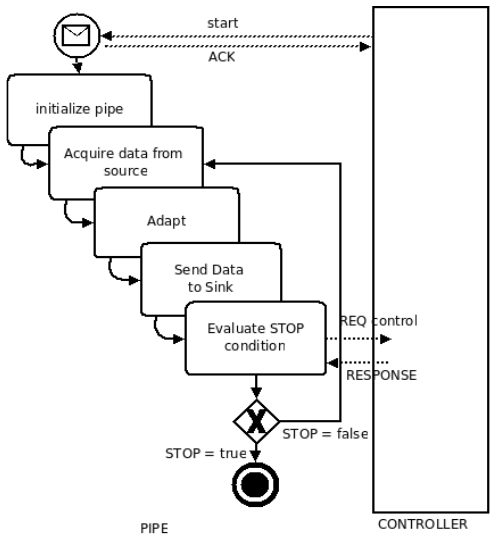


Figure 1. BPMN-like notation for a synchronous pipe from a source to a sink operation.

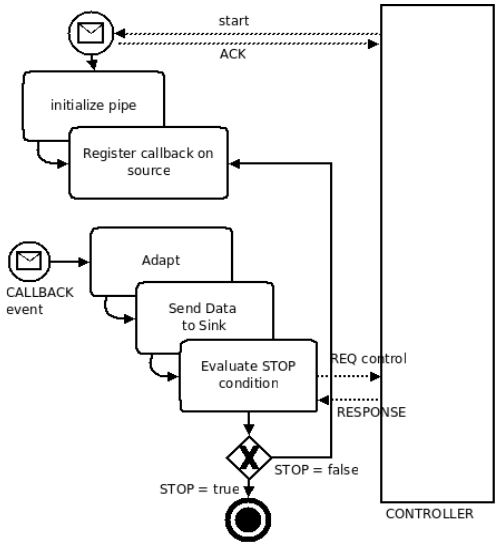


Figure 2. BPMN-like notation of an asynchronous pipe between a source and a sink. The callback endpoint is invoked when data can be consumed.

The difference between the (a) and the (b) template is that in the second the data source asynchronously emits a data and requires that a callback endpoint is registered in order to consume data when data are ready. The different design is depicted in Figs. 1 and 2 where a BPMN [15]-like notation is used instead of showing the XML code, which results too verbose to fit the limit of this article.

## V. PROTOTYPE

The objective of our prototype is to show a living system that allows users to select real objects in a room and to compose them building a real time orchestration starting from the user interaction.

2D barcodes systems like Datamatrix and QR are attached with no cost to any object in order to "augment" their features realizing a virtual connection with a one its digital pair. Appropriate programs can recognize codes and download linked information from the Internet. To implement our point-click-and-compose interactive paradigm, we adopt QR barcodes so the user can point an object and retrieve what that object is able to perform. Given the verbosity even of a simple WSDL document, we choose to encode in the barcode only a URL to reference it. The user points a smart phone against the barcode, then the phone decodes the visual tag and asks an online server to parse the WSDL document to obtain the list of operations. Selecting two different actions from two different objects (or even from the same object) a pipe can be constructed. The WS-BPEL templates are filled with real endpoints, deployed on the WS-BPEL engine and then activated. To implement a prototype we needed some "things" to become endpoints of pipes. Thus, we instrumented normal objects with some SOAP messaging abilities deploying personal computers and notebook to simulate sources and sinks.

## VI. FUTURE WORKS AND CONCLUSION

Capabilities of objects are well expressed with WSDL and translated into human readable lists of actions in the phone user interface. The main advantage is the ability to generate WS-BPEL at runtime and to create new executable processes (the hyperpipes) with the point-select-and-compose interaction.

The overall design results well conceived for the transmission of "data as documents" between different objects while data streaming is less supported by the Web services stack and SOAP is only used for exchanging session descriptions and that commuting to other protocols in the communication stack. The choice to model objects like opaque components able to perform operations poses some issues in the seamless connection with other Web resources. It is clumsy to make a pipe having as endpoint a Web page or a RSS feed because even if these are digital objects, they

 ISBN: 978-1-61208-100-7 

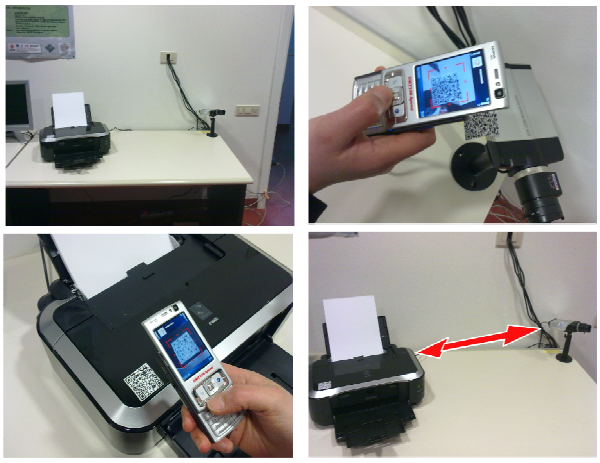

Figure 3. Selecting two different actions from two different objects (or even from the same object) a pipe can be constructed.

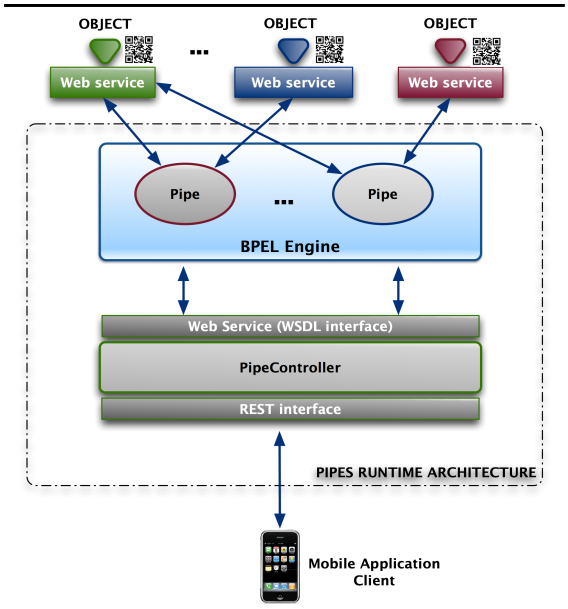


Figure 4. Hyperpipes at runtime. Objects are instrumented by Web services and Pipes are implemented as WS-BPEL processes created and deployed at runtime. A mobile phone is used to point objects in the environment and to retrieve the WSDL specification.

need to be wrapped by a WSDL interface and decorated with a service implementation. The resource-oriented nature of the Web is somehow in contrast with the procedure-oriented architecture of Web services. Some authors [6] consider RESTful the only choice for a Web of things architecture and advocate some motivations related to the programmatic complexity of Web services and according to their experience, not well suited for end-user to create ad-hoc applications. Among the motivations is that discovery of Web services via Universal Description Discovery and Integration (UDDI) is not suitable for sensors or devices because the UDDI-based discovery has not context information (e.g., where a sensor is placed). In our work the problem of discovery is simply by-passed by the fact that services are discovered by users when they are close to an object using some proximity technology (the QR tags in our work) and the programmatic complexity is totally hidden to end-users by the automatic generation of WS-BPEL executables. As mentioned before, we think that orchestrations should include the largest set of element types, both real and virtual, and represented by either stateful processes (WS-*) or stateless resources (REST). One next achievement is to build such a universal orchestration starting from user interactions in the environment.

Regarding the user interaction we conclude that building a pipe between two objects results as a straightforward task. Composing multiple pipes with processor in cascade is somehow less intuitive and requires the user to know how the underlying process is created. The use of QR has revealed to be a practical choice very easy to implement and quite easy for users to manage. Nevertheless, the conclusions on human interaction here reported are merely qualitative and based on the experience of few test users. We plan to make a more accurate usability evaluation in a future work.

## REFERENCES

[1] OASIS Web Services Business Process Execution Language (WS-BPEL) TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel. Last accessed on 01-06-2010

[2] QuadraSpace. http://www.quadraspace.org/. Last accessed on 01-06-2010

[3] Carboni, D. and Zanarini, P. Wireless wires: let the user build the ubiquitous computer. Proceedings of the 6th international conference on Mobile and ubiquitous multimedia, (2007), 169–175.

[4] Deugd, S.D., Carroll, R., Kelly, K., Millett, B., and Ricker, J. SODA: Service Oriented Device Architecture. IEEE Pervasive Computing 5, 2006, 94-96, c3.

[5] Fielding, R.T. Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine, 2000.

[6] Guinard, D., Trifa, V., Pham, T., and Liechti, O. Towards physical mashups in the web of things. Proceedings of INSS, (2009).

[7] Holland, J.H. Emergence: from chaos to order. Addison-Wesley Longman Publishing Co., Inc., 1998.

[8] Jammes, F. and Smit, H. Service-oriented paradigms in industrial automation. IEEE Transactions on Industrial Informatics 1, 1 (2005), 62–70.

[9] Kindberg, T., Barton, J., Morgan, J., et al. People, places, things: web presence for the real world. Mob. Netw. Appl. 7, 5 (2002), 365-376.

[10] Ragget, D. The Web of Things: Extending the Web into the Real World. SOFSEM 2010: Theory and Practice of Computer Science. Springer Berlin / Heidelberg, (2010), 96-107.

[11] Salminen, T., Hosio, S., and Riekki, J. Enhancing Bluetooth Connectivity with RFID. Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications, IEEE Computer Society (2006), 36-41.

[12] Sommer, S., Scholz, A., Buckl, C., et al. Towards the Internet of Things: Integration of Web Services and Field Level Devices. .

[13] de Souza, L.M., Spiess, P., Guinard, D., Kohler, M., Karnouskos, S., and Savio, D. Socrades: A web service based shop floor integration infrastructure. Lecture Notes in Computer Science 4952, (2008), 50.

[14] Trifa, V., Wieland, S., Guinard, D., and Bohnert, T. Design and implementation of a gateway for web-based interaction and management of embedded devices. Submitted to DCOSS, (2009).

[15] White, S.A. Introduction to BPMN. IBM Cooperation, (2004), 2008–029.

 ISBN: 978-1-61208-100-7 