16th IMEKO TC4 Symposium
Exploring New Frontiers of Instrumentation and Methods for Electrical and Electronic Measurements
Sept. 22-24, 2008, Florence, Italy

# A UDDI based distributed measurement system

Fabrizio Ciancetta [1], Giovanni Bucci [1] and Carmine Landi [2]
[1] *Dipartimento di Ingegneria Elettrica e dell'Informazione, Università dell'Aquila*
*Poggio di Roio, 67040 L'Aquila – Italy, Phone +39.0862.434634, Fax +39.0862.434403,*
*Email: {fabrizio.ciancetta, bucci}@ing.univaq.it*
[2] *Dipartimento di Ingegneria dell'Informazione, Seconda Università di Napoli*
*Via Roma 29, 81031 Aversa (CE) – Italy, Phone +39.081.5010239, Fax +39.081.5037042,*
*Email: carmine.landi@unina2.it*

***Abstract-*** In this paper the authors are mostly concerned with the implementation of a measurement system specially suited for widely distributed applications. The system is based on smart web sensors, where the Internet links together the individual sensing elements. The system adopts the UDDI (Universal Description, Discovery, and Integration) registry, a de facto standard for managing the web services on the web. The proposed solution allows having, at all times, an overview of the sensors and of the available services. Thus, when a new sensor is inserted in the network, its services and its functionality are automatically added into the registry without reconfiguring the net. The sensor is presented in a taxonomy of distributed measurement systems for metrological monitoring applications with a standard interface via a tModel. Measurement results supplied by the sensors are made available through a web service, so that all users can build up their own applications. Some preliminary results of the proposed distributed platform are showed for the monitoring of environmental parameters.

## I. Introduction

Several applications require for a distributed measurement system able to measure the same or different parameters at different points in a wide area. In recent years, wide varieties of solutions have been proposed for remote measurement and data transmission. Distributed systems based on smart web sensors represent the best solution to many different measurement problems [1,2,3]. By adopting these sensors, a client can receive the measure of a particular physical quantity with a browser or an application, developed to receive information from the web server.
From developer side, smart web sensors present always a closed approach to interact with them, so web services are adopted in order to give a standard approach in developing a Service Oriented Architecture. From network side, Internet is a widely adopted network where any user is uniquely identified with its IP address. So, to implement a distributed measurement system, it is necessary to use a common and open communication protocol to exchange information and a methodology to auto-configure any smart sensor is linked to the network. During the development of a smart Web sensors network, two are the main problems to solve: i) how "discover" the smart web sensor and ii) which interface is "published" to consume the services. Our proposal is to use the UDDI technology.
The original vision for UDDI was as standards and tools that would help companies conduct business with each other in an automated fashion. The searchable central registries provide a publish and subscribe mechanism to store the agency and service descriptions and to point to detailed technical specifications that define the interfaces to these services. The UDDI registry is a possible solution of the previously introduced measurement problems, because it provides a standardized method for publishing and discovering information about web services and because it implements a customization of type's description that can be used to standardize the smart web sensor interface to the network.
Our aim is to make possible the use of this technology to share measurement information supplied by smart Web sensors with the same straightforwardness and reduced costs.
This paper describes the development of the distributed platform. Additionally, results from some preliminary tests are presented to analyse the performance of the system.

## II. Network architecture of distributed measurement system

The central elements of the proposed network are the UDDI registry, the Web service UDDI access and the smart UDDI sensor. The UDDI registry is the load-bearing wall of the whole network. It is composed by a personal computer in which a UDDI registry has been implemented.
The smart UDDI sensor is a smart Web sensor which both publishes on Internet a Web service of application-depending functionalities and publishes and stores in the UDDI registry its functionalities.
The Web service UDDI access is a Web service with three distinct functionalities: first of all it allows at a client to access the network via SOAP messages, second it interfaces the client to the UDDI registry to search a particular smart Web sensor and third it interfaces the client to the smart UDDI

sensor to consume the Web service. A relational diagram of the proposed architecture is reported in the Figure 1, while in the next sections the main parts of the network will be explained in more details.

### A. Taxonomies of Smart UDDI Sensors

In the proposed network, every smart UDDI sensor (SUS) publishes its services both into the Internet network and into the UDDI registry, as reported in Figure 1. For the Internet publishing part, the SUS presents a Web service interface allowing every user to access to its functionality as in [1, 4, 5]. For the UDDI registry publishing part, the smart Web sensor stores in a UDDI record all the standard information (such as company name, business identifier, etc...) and technical information (such as tModel) to allow every consumer to search and to find the specific SUS [6].
However, every SUS can provide a different list of services and a different use of the same services caused by a different interface that the smart Web sensor present to the consumer. In the proposed system, a tModel describing the sensor services has been developed as a basic common interface from which every smart Web sensor realizes a concrete implementation [7]. In this way, a taxonomy of smart Web sensors can be developed to create a sensors network [8] that presents the same interface to the client even if they have a different implementation [9].

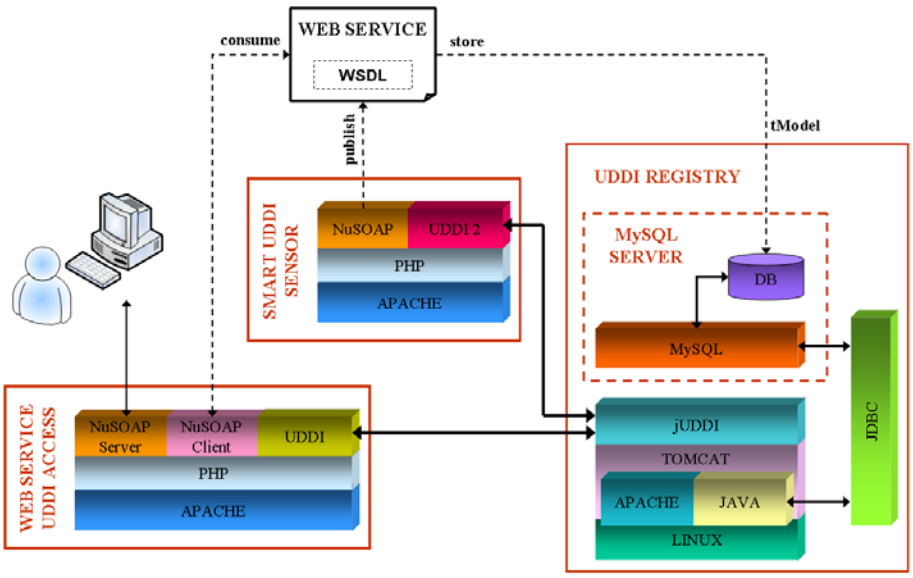


Figure 1. Network and software architecture of the proposed system

### B. Web Service UDDI Access

UDDI adopts a standard protocol based on SOAP to send and receive information [10]. The access to the UDDI registry is defined by the APIs reported in [11]. To give much more freedom to the client that uses the distributed measurement system, a special Web service has been developed to consume the network of SUSs, accessing to the UDDI registry [12]. As reported in Figure 1, the Web service UDDI access (in short SWUA) provides an interface that links the client to the UDDI registry and to the network of SUSs. For the UDDI interface it publishes only the methods that allow the discovering of a SUS, while, for the SUS interface, it publishes some generic method to consume the Web service of SUS involved in the search. Ultimately, the Web service UDDI access provides a complete access to the network of SUSs with some simple tasks: i) the client sends a query to the WSUA using a Web method; ii) the query is forwarded by the WSUA to the UDDI registry using the UDDI APIs; iii) the response of the query is packaged in a software object with many information about the STUs involved in the search; iv) the client can access to the Web service of SUS utilising another WSUA's Web method; v) the WSUA sends a SOAP message to the STU and waits a response that sends back to the client.

## III. The Developed Distributed Measurement System

### A. The developed Web Service UDDI Access

A personal UDDI registry has been implemented to develop the distributed measurement system. It has been used jUDDI project ver.2.0 because it is an open source Java implementation of the UDDI specification for Web services. To implement the UDDI registry by jUDDI, we utilised a local personal computer with Fedora Core 4, where we installed: i) Java 2 SDK - used Sun's Java 2 SDK SE, version

1.4.2_04; ii) J2EE build and runtime environment - used Sun Java 2 Enterprise Edition (J2EE) 1.3.1; iii) Web server and/or servlet container - used Apache Tomcat, version 5.0.24; iv) SOAP processing framework - used the version of Apache Axis that ships with jUDDI; v) Data storage mechanism - used the MySQL relational database, version 4.0.19; vi) UDDI registry framework - the registry framework jUDDI.

### B. The developed smart UDDI sensor

The hardware architecture is based on a motherboard that is made operative by a light Linux distribution. The distribution has been installed on a USB pen driver, while the database is stored in an other USB mass storage drive to separate the kernel with the data. A diagram block of the whole system is reported in the Figure 2-3.

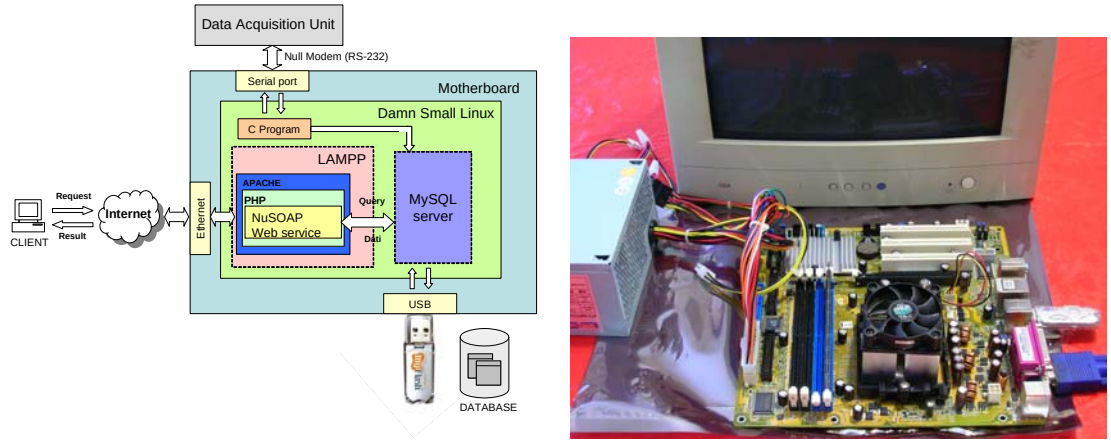


Figure 2 - 3. Software and hardware architecture of smart Web sensor

The smart UDDI sensor is an embedded system, able to perform measurement tasks and to communicate over the Internet network. The distributed measurement system has been implemented adopting only open source software. As a multipurpose operating system, Linux is used for a wide variety of purposes including networking, software development, as well as an end-user platform. Apache is an implementation of an HTTP server and it is the most popular Web server in use today. MySQL is an implementation of a database server that is known for its speed and reliability. PHP is general-purpose scripting language that is particularly suited to Internet-based system development and is the most widely used Apache module.
The proposed smart Web sensor has been adopted for an environmental monitoring application. In this system the Linux embedded board performs some tasks: i) it receives information from a data acquisition unit via the RS-232 port; ii) the C program saves the received data in the database mounted on the USB mass storage device; iii) the Web server manages a PHP Web service for the publishing on demand of the present services through the network interfaces connected to the system; iv) it communicates with the UDDI registry for the discovering of the services; v) it communicates with the client part of the WSUA.
The data acquisition unit is an embedded system based on a PIC 18F452 connected to the motherboard via a RS-232 interface. A low level program, written in C, performs same tasks: it manages the acquisition timing (10 seconds), it queries the data acquisition unit to perform a measure; it parses the string which contains the measure in a floating number; it saves the new data with the time stamp in the daily table; it performs the evaluation of maximum, medium and minimum at every new value so, when the pointer of current value has performed a complete cycle in the daily table, the data can be saved in the historic table.

### C. The new UDDI class

In literature exist many implementation of open source and cross platform classes based on Java and PEAR packages (a framework and distribution system for reusable PHP components) to develop application software to access UDDI registry. However, these classes present the limitation of performing only searching and querying on UDDI registry but they do not allow any new registration, data storing and access because not all the UDDI APIs and the management of authorization token have been implemented.
Starting from the UDDI class based on PEAR package, a new class (called UDDI2) has been

developed to perform the registration and to manage all the information of storing in the UDDI registry. The new methods added to the existing UDDI class are:

- get authToken: when the SUS needs to perform an access to the UDDI registry, it logs itself and requires an authorization Token;
- discard authToken: when the SUS ends the operations of saving services into the UDDI registry, it discards the authorization Token to perform a new access;
- save service/tModel: with these two methods the SUS can save the service and the tModel into the UDDI registry;
- delete service/tModel: with these two methods the SUS can delete the service and the tModel into the UDDI registry;

## IV. THE PROPOSED UDDI STRUCTURE FOR THE SMART UDDI SENSOR

### A. The proposed taxonomies of developers

The first element to store in the UDDI registry is the *BusinessEntity* that reports the entity that provides the services. In this case, the entity is the smart Web sensor. After some information about the developer, the *BusinessEntity* has been referenced to D-U-N-S and to a tModel structure that defines the development group that develops the smart Web sensor via the item *keyedReference*. A new tModel has been developed which describes the measurement group who has developed the smart Web sensor. It references to another tModel that describes the department to which the measurement group belongs. Using a tree-referencing system, the developed smart Web sensor can be referenced to many entities that are grouped using the item *categoryBag* and are distinguished via the *keyValue* attributes present in the item *keyedReference*. The smart Web sensor is the leaf of a tree of references that allow creating a taxonomies of groupages. During a query, this system allows to categorize the resources and to provide results in according to the groupages. So, every level is a taxonomies.

### B. The proposed taxonomies of services

Every smart Web sensor is a remote measurement system that provides some measurement services. The entity alike in the UDDI registry that lists all the services and the functionalities provided by a *BusinessEntity* is the *businessService*. The Figure 4 reports the developed *businessService*.

```
<businessService
  serviceKey="d5921160-3e16-11d5-98bf-002035229c64"
  businessKey="0076b468-eb27-42e5-ac09-9955cff462a3"">
  <name>Services present on measurement station</name>
  <description xml:lang="en">Daily and Historical data </description>
  <bindingTemplates>
    <bindingTemplate
      serviceKey="d5921160-3e16-11d5-98bf-002035229c64"
      bindingKey="3f2a1343-124f-1237-63c1-2924d12ab215">
      <description xml:lang="en">
        SOAP binding for daily service
      </description>
      <accessPoint URLType="http">
        http://192.168.0.4/services/Daily.php
      </accessPoint>
      <tModelInstanceDetails>
        <tModelInstanceInfo
        tModelKey="uuid:87e84e70-11ab-11db-8e70-85bdac8c5d57" />
      </tModelInstanceDetails>
    </bindingTemplate>
    <bindingTemplate
      serviceKey="554313bd-12b1-22b1-7649-22751963cd60"
      bindingKey="5254fa21-cb21-45ef-3215-0042222ba23c">
      <description xml:lang="en">
        SOAP binding for Historic serice
      </description>
      <accessPoint URLType="http">
        http://192.168.0.4/services/Historic.php
      </accessPoint>
      <tModelInstanceDetails>
        <tModelInstanceInfo
        tModelKey="uuid:977b94ad-11ab-11db-94a0-be654348e44a" />
      </tModelInstanceDetails>
    </bindingTemplate>
  </bindingTemplates>
</businessService>
```

Figure 4. *businessService* of smart Web sensor

The main important things to underline are:

1. the item *businessKey* provides the link between the *BusinessEntity* and the *businessService*.
2. the item *bindingTemplates* contains a collection of items *bindingTemplate* which describe the services present on the service provider (smartWeb sensor).
3. the item *tModelInstanceDetails* references to the tModel which describes the software interface of the service. The tModel represents a “contract” between the service provider and the interface of the Web services.
4. the item *accessPoint* addresses the URL of the Web services.

The tModels of Web services are reported in the Figure 5 and in the Figure 6. In these listings the main important items are the item *overviewURL* that addresses to the tModel URL and the item *categoryBag* that categorizes the *tModels* to create another mechanism of taxonomies of Web service.

```
<tModel
  tModelKey="uuid:87e84e70-11ab-11db-8e70-85bdac8c5d57"
  operator="Fabrizio Ciancetta" authorizedName="fabcian">
  <name>Daily service </name>
  <description xml:lang="en">Daily service present on
     measurement station interface</description>
  <overviewDoc>
    <description xml:lang="en">wsdl link</description>
    <overviewURL>
      http://measurement.ing.univaq.it/tmodels/Daily.wsdl
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
      tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
      keyName="uddi-org:types" keyValue="wsdlSpec" />
    <keyedReference
      tModelKey="uuid:0e822af2-66ef-29b1-83a7-117ee1b79121"
      keyName=" measurement " keyValue=" Daily " />
  </categoryBag>
</tModel>
```

Figure 5. tModel of Web service Daily Service

```
<tModel
  tModelKey="uuid:977b94ad-11ab-11db-94a0-be654348e44a"
  operator="Fabrizio Ciancetta" authorizedName="fabcian">
  <name>Historic service </name>
  <description xml:lang="en">Historic service present on
     measurement station interface</description>
  <overviewDoc>
    <description xml:lang="en">wsdl link</description>
    <overviewURL>
      http://measurement.ing.univaq.it/tmodels/Historic.wsdl
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
      tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
      keyName="uddi-org:types" keyValue="wsdlSpec" />
    <keyedReference
      tModelKey="uuid:0e822af2-66ef-29b1-83a7-117ee1b79121"
      keyName=" measurement " keyValue=" Historic " />
  </categoryBag>
</tModel>
```

Figure 6. tModel of Web service Historic Service

### C. The proposed interface

For the proposed smart Web sensor we developed a UDDI interface to standardize the input format, the output format and the methods [3, 16]. Every format is stored in a UDDI tModel. In particular, two Web methods were developed:

1. GetDailyService(service,start_time,stop_time): with this method a client can request to download the daily data from the start time to the stop time from the smart Web sensor . The result is an array of items that presents the stored value and the time when the event is occurred.
2. GetHistoricService(service,start_date,stop_date): with this method a client can request to download the historic data from the start date to the stop date from the smart Web sensor . The smart Web sensor provides some processing before storing the data into the local database. In particular, it calculates the Maximum, Medium and Minimum values. The result is an array of items in which every item is a collection of three sub-items. Every sub-item presents the Maximum, Medium or Minimum value and the date and time when the event is occurred.

## V. EXPERIEMENTAL RESULTS

The whole system has been tested to verify the proposed architecture. Using the Web service UDDI access, a client has been developed with Visual Studio 2005 .NET to consume the proposed smart UDDI sensor. The client has been written in VB.NET using the Framework .NET 2.0.
The Windows form, shown in Figure 7, allows the user to access to the smart Web sensor and, in particular, to download the data from the daily table and the historic table. Selecting the period of time from the ActiveX control and push on the Download Data, the Windows form access to the WSUA which download the data for the SUS.

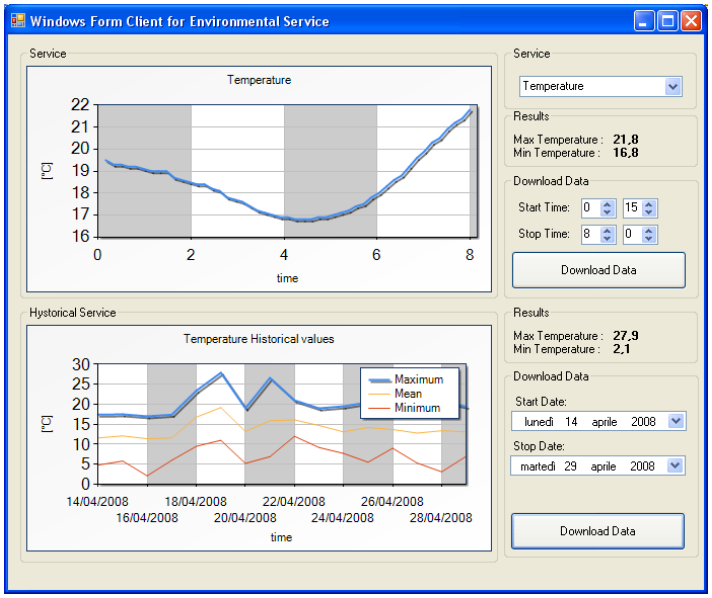


Figure 7. A screenshot of Client GUI

## VI. CONCLUSIONS

In this paper a new architecture of smart Web sensor based on UDDI registry is presented. The architecture adopts the tModel description to create a common interface to publish the services and to create taxonomies of developer and taxonomies of services. In this way, a client, accessing into the UDDI registry, can found the Web service performing *BusinessEntity* queries (smart Web service), *businessService* queries (services) and tModel queries (interface or taxonomies).
A smart Web sensor was developed using only open source and freeware software development tools. In particular, we developed a measurement system based on embedded Linux distribution mounted in a USB pen drive. In this way, a motherboard, a CPU, some memory and a USB with embedded Linux are enough to realize a complete measurement system.
The system was adopted to develop an environmental application. The acquisition part was developed adopting a PIC 18F542, connected with the motherboard via the RS-232 interface. To give much more freedom to the whole system, a new Web service for UDDI access was developed, to create an interface between the client and the UDDI registry and between the client and the smart Web sensor. In this way the client can see a Web service of Web services and can found the specific Web service only with standard Web method without knowing UDDI APIs. The results show that the client can route the request till the developed smart Web sensor and can download the data from its embedded database.

## References

[1] F. Ciancetta, B. D'Apice, D. Gallo, C. Landi, "Architecture for Distributed Monitoring based on Smart Sensor and Web Service", in Proc. of *IEEE Instrumentation and Measurement Technology Conference Sorrento*, Italy, 24-27 April 2006.
[2] F. Ciancetta, E. Fiorucci, B. D'Apice, C. Landi, "A Peer-to-Peer Distributed System for Multipoint Measurement Techniques", in Proc. of *IEEE Instrumentation and Measurement Technology Conference*, Warsaw, Poland, May 1-3, 2007
[3] Kang Lee and Eugene Song, "Smart Transducer Web Services Based on the Proposed IEEE 1451.0 Standard", in Proc. of *IEEE Instrumentation and Measurement Technology Conference*, Warsaw, Poland, May 1-3, 2007.
[4] Tomasz Mielcarz and Wieslaw Winiecki, "The Use of Web-services for Development of Distributed Measurement Systems", in Proc. of *IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, 5-7 September 2005 ,Sofia, Bulgaria, pp 320–324.
[5] F. Ciancetta, B. D'Apice, D. Gallo, C. Landi, "Plug-n-Play Smart Sensor Based on Web Service", *IEEE Sensors Journal*, 7, issue 5, May 2007, pp 882–889.
[6] Jiamao Liu, Ning Gu, Yuwei Zong, Zhigang Ding, Shaohua Zhang and Quan Zhang, "Service Registration and Discovery in a Domain-Oriented UDDI Registry", in Proc. of *the Fifth International Conference on Computer and Information Technology (CIT'05)*, Shanghai, China, 21-23 September, 2005, pp 276–283.
[7] OASIS, *UDDI Registry tModels, Version 2.04*, Available at http://uddi.org/taxonomies/ UDDI_Registry_tModels.htm
[8] S.M. Pahlevi, H. Kitagawa, "Taxonomy-based adaptive Web search method", in Proc. of *International Conference on Information Technology: Coding and Computing*, Las Vegas, NV, USA, 8-10 April, 2002, pp 320–325.
[9] OASIS, *Providing a Taxonomy for Use in UDDI Version 2*, Available at http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-taxonomy-provider-v100-20010717.htm
[10] OASIS, *UDDI Spec Technical Committee Draft, Dated 20041019*, Available at http://www.oasis-open.org/committees/uddi-spec/doc/spec /v3/uddiv3.0.2-20041019.htm.
[11] OASIS, *UDDI Version 2.04 API*, Published Specification, Dated 19 July 2002 Available at http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf.
[12] Jian Jun Yu, Gang Zhou, "Dynamic Web Service Invocation Based on UDDI", in Proc. of *IEEE International Conference on E-Commerce Technology for Dynamic E-Business (CEC-East'04)*, Beijing, China,13-15 September, 2004, pp 154–157.