Domotics Lab

Una grammatica XML per la Domotica

Fabio Testa

Vittorio Miori

CONSIGLIO NAZIONALE
DELLE RICERCHE

Indice

1 Una grammatica XML per la Domotica

XML è un metalinguaggio che fornisce un insieme standard di regole sintattiche per modellare la struttura di documenti e dati. Questo insieme di regole, dette più propriamente specifiche, definiscono le modalità secondo cui è possibile crearsi un proprio linguaggio di markup.

L'acronimo XML sta per *eXtensible Markup Language*.

Lo standard XML è un formato testuale, simile per molti aspetti al linguaggio HTML, ideato specificamente per memorizzare e trasmettere dati.

Un'origine XML è costituita da elementi XML, ciascuno dei quali formato da una tag iniziale (<nometag>), una tag finale (</nometag>) e da informazioni, dette contenuto, comprese tra le due tag. Analogamente a un documento HTML, un documento XML contiene del testo annotato con tag.

Tuttavia, diversamente dal linguaggio HTML, lo standard XML consente di includere una numero illimitato di tag, ciascuna indicante non l'aspetto, ma il significato di un elemento testuale.

Ad esempio, un elemento XML può essere contrassegnato con una tag come tecnologia, numero del prodotto o come nome. È compito dell'autore del documento stabilire il tipo di dati da utilizzare e i nomi delle tag che meglio si adattano ad essi.

Nel linguaggio XML un documento può essere accompagnato da file XSD (*Xml Schema Definition*), il quale in sostanza ne definisce le regole, quali l'indicazione degli elementi presenti e la relazione strutturale intercorrente tra di essi.

I file XSD aiutano a convalidare i dati quando l'applicazione ricevente

non dispone di una descrizione incorporata dei dati in arrivo. In ogni caso, tali file sono facoltativi con l'XML.

I dati inviati insieme ad un file XSD, sono dati XML validi (*valid XML*). In tal caso, un *parser* XML controlla i dati in arrivo sulla base delle regole definite nel file XSD al fine di accertare la corretta strutturazione dei dati.

I dati inviati senza un file XSD sono noti come dati *ben formati* (*well-formed*). In ambedue i casi, i dati codificati in formato XML sono autodescrittivi in quanto sono inserite delle tag descrittive tra i dati. Il formato flessibile e aperto utilizzato dal linguaggio XML ne consente l'impiego dovunque vi sia bisogno di scambiare o trasferire informazioni. Tutto ciò lo rende estremamente potente. Utilizzando uno schema, è quindi possibile definire con precisione i nomi di elementi permessi in un documento e, all'interno di determinati elementi, quali sottoelementi, attributi e relazioni sono consentiti. Un autore può importare frammenti da altri schemi ed estendere i tipi ereditandoli. Ciò consente relazioni complesse tra gli elementi, pur conservando la semplicità di una struttura lessicale ad albero.

XML quindi, grazie ad una *grammatica XML Schema* (il file XSD di qui si è parlato in precedenza), diventa un linguaggio per definire nuovi linguaggi.

Sfruttando questo potente strumento è stato quindi pensato domoML, un linguaggio costituito da tags specificamente utili per applicazioni in ambito domotico, con l'obiettivo di creare appunto una lingua comune tra i vari standard.

Addrentiamoci ora nei dettagli della realizzazione dei concetti finora descritti.

1.1 XML Schema

L'XML Schema è l'unico linguaggio di descrizione del contenuto di un file XML che abbia per ora raggiunto la validazione ufficiale del W3C[1].

Come tutti i linguaggi di descrizione del contenuto XML, il suo scopo è delineare quali elementi sono permessi, quali tipi di dati sono ad essi associati e quale relazione gerarchica hanno fra loro gli elementi contenuti in un file XML. Ciò permette principalmente la validazione del file XML, ovvero la verifica che i suoi elementi siano in accordo con la descrizione in linguaggio XML Schema.

Un'altra cosa che XML Schema permette è l'estrazione, o meglio una visione, da un file XML di un insieme di oggetti con determinati attributi ed una struttura.

Un XML Schema Definition (XSD) è un esempio concreto (*instance*) di schema XML scritto in linguaggio XML Schema. Un XSD definisce il tipo di un documento XML in termini di vincoli: quali elementi ed attributi possono apparire, in quale relazione reciproca, quale tipo di dati può contenere, ed altro. Può essere usato anche con un programma di validazione, al fine di accertare a quale tipo appartiene un determinato documento XML.

1.2 Il linguaggio domoML

Nella definizione di un linguaggio atto ad implementare l'interoperabilita' tra standard domotici molto diversi, è stato considerato imperativo porre l'attenzione in primis ad un approccio il più generale possibile.

[1] http://www.w3.org

Questo è assolutamente necessario affinchè il linguaggio stesso possa essere utilizzato, non solo nel più vasto campo di applicazioni possibili, ma anche come linea guida per il miglioramento delle funzionalità realmente portabili verso l'interoperabilità.

Lo scopo primario nel definire questo nuovo linguaggio, è proprio spingere i produttori a rendere facilmente reperibile il massimo numero di informazioni che i dispositivi dovrebbero essere in grado di fornire a qualsiasi entità interagente.

L'applicativo domoNET, un applicativo sviluppato dal Laboratorio di Domotica per dimostrare l'effettiva interoperabilità tra middleware domotici esistenti [2], allo stato attuale, è costretto ad inferire la tipologia di un dispositivo da alcune sue caratteristiche, mentre sarebbe auspicabile che fosse il dispositivo stesso a comunicare prima di tutto alcune informazioni generali sulle sue funzionalità, come ad esempio l'essere una lampadina o in generale l'essere un dispositivo di illuminazione.

In questo modo non solo risulterebbe molto più facile per domoNET associare dispositivi di standard domotici diversi per categorie di funzionalità, così da poter, per esempio, accendere tutte le luci da un unico pannello o pulsante, ma la stessa interfaccia utente di un qualsiasi controller risulterebbe molto più intuitiva e accessibile all'utente medio, per il quale è necessario astrarre il più le informazioni da gestire.

Esempio principe è la funzionalità acceso/spento di un qualsiasi dispositivo; non essendo presente un nome universalmente utilizzato per questo servizio (Power , SetPower, Set_Power, ON/OFF sono solo alcuni dei no-

[2] http://sourceforge.net/projects/domonet

mi riscontrati), e tenendo presente la generalità come principio base nella costruzione dell'interfaccia utente da implementare, che impedisce di utilizzare un'enumerazione finita di possibili stringhe, risulta impossibile fornire all'utente un semplice bottone per accendere il dispositivo, piuttosto che un campo testo dove inserire un valore binario o booleano che ovviamente è molto meno immediato e intuitivo.

Ecco quindi le principali problematiche che attualmente domoML vorrebbe risolvere. Porsi come lingua comune tra i diversi standard potrebbe quantomeno ampliare le vedute sulle problematiche che si presentano nel rendere interoperabili diversi middleware domotici, tenendo oltretutto presente che la storia dell'informatica insegna che una qualsiasi tecnologia, se punta ad una diffusione su larga scala, deve avere uno standard unico e universalmente accettato.

All'atto pratico domoML è un un linguaggio per rappresentare in maniera compatta i dispositivi domotici: i relativi servizi offerti e le relative interazioni, astraendo dalla tecnologia di appartenenza. domoML, usato con questa architettura, è un *middle language* da e verso quale tradurre le rappresentazioni dei dispositivi e dei pacchetti.

Tutta la parte logica dell'architettura usa domoML e solamente le interazioni fisiche con i dispositivi per costruire i corrispondenti *domoDevice* e per l'esecuzione dei servizi, all'interno dei moduli (techmanager), usano il linguaggio specifico proprio di ogni tecnologia.

domoML si occupa attraverso i tag:

- domoDevice:

alla descrizione astratta dei dispositivi fatte secondo domoML;

- domoMessage:

  alla descrizione astratta delle interazioni da e verso i dispositivi fatte secondo domoML.

1.2.1 domoDevice XML Schema

Il domoDevice ha lo scopo di creare un meccanismo semplice, compatto ed efficace per rappresentare i dispositivi astraendo dalla tecnologie.

Un domoDevice può essere rappresentato attraverso un albero largo e poco profondo (la grammatica è altamente attributata e con pochi figli). L'albero ha al massimo 4 livelli:

1. device:

   apre il tag che descrive il domoDevice dando generiche informazioni come:

   - description:

     una descrizione in lingua naturale del domoDevice;

   - id:

     valore usato per prendere e generare il domoDeviceId e che dentifica il domoDevice all'interno del web service;

   - manufacturer:

     il produttore del dispositivo;

   - positionDescription:

     una descrizione in lingua naturale della locazione del dispositivo;

- serialNumber:

  il numero seriale (può essere composto anche da lettere) del dispositivo;

- tech:

  la tecnologia originale del domoDevice rappresentato;

- type:

  la tipologia del dispositivo(illuminazione, sicurezza, entertainment, ecc.);

- URL:

  usato per prendere e generare il domoDeviceId. Identifica il web service che tiene di dispositivo.

Tutti i campi sono opzionali eccetto id, URL e tech perché permettono di identificare e usare il dispositivo.

2. service:

   descrive un singolo servizio offerto dal domoDevice. Questa informazione è usata per creare un domoMessage e per convertirlo nel messaggio proprio della tecnologia del dispositivo(tech message). Per ogni domoDevice ci possono essere più servizi. I campi di questo tag sono:

   - description:

     una descrizione in linguaggio naturale del servizio;

   - name:

     un identificatore che può essere utile quando il domoMessage viene generato e tradotto in tech message.

- output:

  il domoML.domoDevice.DomoDevice.DataType si aspetta un valore di ritorno quando viene eseguito il domoMessage. Questo campo non va messo se non è previsto alcun valore di ritorno;

- outputDescription:

  una descrizione in linguaggio naturare per l'attributo output. Se non è presente l'attributo output neanche questo attributo deve essere messo;

- prettyName:

  una etichetta in in linguaggio naturale del servizio;

3. due possibili tag per questo livello:

   - input:

     indica che il servizio ha bisogno di un valore di input per poter essere eseguito. Questo tag è opzionale e ogni servizio può avere più input. I campi per questo tag sono:

     – name:

       un identificativo dell'input;

     – description:

       una descrizione dell'input;

     – type:

       il domoML.domoDevice.DomoDevice.DataType dell'input atteso;

   - linkedService:

il servizio da associare quando questo servizio viene eseguito. Può essere invocato un servizio dello stesso o di un qualsiasi altro domoDevice. L'associazione di servizi non ha vincoli semantici ovvero è possibile combinare un servizio con qualsiasi altro servizio a patto di riuscire a dare dei valori ragionevoli ai linkedInput(discusso successivamente). Questo tag è il cuore di domoML in quanto permette di risolvere il problema della cooperazione tra dispositivi reali eterogenei. Questo tag è opzionale. I suoi attributi sono:

- URL:
  l'identificativo del web service che gestisce il domoDevice da collegare al servizio;
- id:
  l'identificativo del domoDevice da collegare al servizio all'interno del web service;
- service:
  il nome del servizio del domoDevice individuato dai precedenti attributi da collegare;

4. in base al tag precedente, a questo livello è possibile avere:

   - allowed:
     se il tag precedente è input. Rappresenta un possibile valore che può assumere l'input. Questo parametro è opzionale e un input può avere più allowed; l'unico attributo per questo tag è:

  - value:

    il valore in formato stringa ammesso.

- linkedInput:

  se il tag precedente è linkedService. Rappresenta l'associazione di un input del presente servizio con un input del servizio da richiamare. Ogni input del servizio da richiamare deve avere un'associazione. In questo modo, il valore dell'input del presente servizio viene passato come input al servizio da richiamare; gli attributi per questo tag sono:

  - from:

    il nome dell'input dal quale prendere il valore;

  - to:

    il nome dell'input che deve usare il valore preso;

  Un esempio di domoDevice per una semplice lampadina è:

```
<device description="lampada a risparimo energetico"
id="0" manufacturer="pholips"
postionDescription="comodino accanto al letto"
serialNumber="xxxxxxxxx" tech="KNX" type="lampada"
url="http://www.questowebservice.it/servizio">
<service description="Get the status"
output="BOOLEAN" outputDescription="The value"
name="GET_STATUS" prettyName="Get status" />
<service description="Set the status"
name="SET_STATUS" prettyName="Set status">
```

```
<input description="The value" name="status"
type="BOOLEAN">
<allowed value="TRUE" />
<allowed value="FALSE" />
</input>
<linkedService id="3" service="setPower"
url="http://www.altrowebservice.it/servizio">
<linkedInput from="status" to="power" />
</linkedService>
</service>
</device>
```

In questo esempio è possibile vedere che il domoDevice descritto è una lampadina a risparmio energetico prodotta dalla ditta *pholips* e che si trova sul comodino accanto al letto. La tecnologia a cui appartiene è *Konnex* e fa parte della categoria *lampada*. La lampada è comandata dal web service *http://www.questowebservice.it/servizio*. I servizi offerti dalla lampada sono due: prendere il suo stato corrente in formato booleano e settare il suo stato attraverso un input, sempre booleano, che può assumere i valori *TRUE* o *FALSE*. Quando viene invocato quest'ultimo servizio, ne viene chiamato anche un altro di nome *setPower* appartenente al domoDevice gestito dal web service *http://www.altrowebservice.it/servizio* con *id* **3**. Il valore dell'input assegnato al primo servizio (status) viene assegnato anche al secondo (power). Se viene invocato quindi il servizio *setStatus* del domoDevice con *url=http://www.questowebservice.it/servizio* e con *id=0* passando come val-

ore di input *TRUE*, viene anche invocato il servizio *setPower* del domoDevice *url=http://www.altrowebservice.it/servizio*, con *id=3* e con valore di input *TRUE*.

Qui di seguito la formalizzazione della grammatica XML (XML Schema) per generare un **domoDevice** corretto (well-formed).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="quali
  <xs:element name="device">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="service"/>
      </xs:sequence>
      <xs:attribute name="description" use="required"/>
      <xs:attribute name="id" use="required" type="xs:string"/>
      <xs:attribute name="manufacturer" use="required" type="xs:string"/>
      <xs:attribute name="postionDescription" use="required"/>
      <xs:attribute name="serialNumber" use="required" type="xs:string"/>
      <xs:attribute name="tech" use="required" type="xs:string"/>
      <xs:attribute name="type" use="required" type="xs:string"/>
      <xs:attribute name="url" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="service">
    <xs:complexType>
      <xs:sequence minOccurs="0">
```

```
        <xs:element ref="input"/>
        <xs:element ref="linkedService"/>
      </xs:sequence>
      <xs:attribute name="description" use="required"/>
      <xs:attribute name="name" use="required" type="xs:string"/>
      <xs:attribute name="output" type="xs:string"/>
      <xs:attribute name="outputDescription"/>
      <xs:attribute name="prettyName" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="input">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="allowed"/>
      </xs:sequence>
      <xs:attribute name="description" use="required"/>
      <xs:attribute name="name" use="required" type="xs:string"/>
      <xs:attribute name="type" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="allowed">
    <xs:complexType>
      <xs:attribute name="value" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
```

```
  <xs:element name="linkedService">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="linkedInput"/>
      </xs:sequence>
      <xs:attribute name="id" use="required" type="xs:string"/>
      <xs:attribute name="service" use="required" type="xs:string"/>
      <xs:attribute name="url" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="linkedInput">
    <xs:complexType>
      <xs:attribute name="from" use="required" type="xs:string"/>
      <xs:attribute name="to" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

1.2.2 domoMessage XML Schema

Il domoMessage ha lo scopo di creare un meccanismo semplice, compatto ed efficace per rappresentare le interazioni tra i domoDevice astraendo dalle tecnologie.

Anche il domoMessage può essere rappresentato attraverso un albero largo e poco profondo (si usa una grammatica altamente attributata e con pochi figli). L'albero ha al massimo 2 livelli:

1. message:

   apre il tag che descrive il domoMessage e contiene i seguenti attributi:

   - message:

     il corpo del messaggio, tipicamente il nome del servizio da invocare;

   - messageType:

     il tipo di messaggio. Può essere:

     – COMMAND:

       se trattasi di servizio da eseguire;

     – SUCCESS:

       se richiesto l'esito successivo al COMMAND. In questo caso il campo message può contenere il valore di risposta dell'esecuzione del servizio;

     – FAILURE:

       se richiesto l'esito successivo al COMMAND. In questo caso il campo message può contenere il codice di errore del fallimento;

   - receiverId:

     l'identificatore del domoDevice ricevente del messaggio;

   - receiverURL:

     l'url del web service che contiene il domoDevice richiesto;

   - senderId:

     l'identificatore del domoDevice mittente del messaggio;

- senderURL:

  l'url del web service che contiene il domoDevice che invia il messaggio;

2. input:

   i valori di input da associare al message. Questo tag è opzionale e ogni tag message può avere più tag input. Gli attributi per questo tag sono:

   - name:

     il nome dell'input;

   - type:

     il *DomoDevice.DataType* dell'input;

   - value:

     il valore in formato stringa dell'input;

Un esempio di domoMessage per accendere una semplice lampadina è:

```
<message message="SET_STATUS" messageType="COMMAND"
receiverId="1"
receiverURL="http://www.altrowebservice.it/servizio"
senderId="0"
senderURL="http://www.questowebservice.it/servizio">
<input name="status" type="BOOLEAN"
value="TRUE" />
</message>
```

In questo esempio viene richisto di eseguire il servizio SET_STATUS sul domoDevice con web service *http://www.questowebservice.it/servizio* e *id* **1**

con input di tipo booleano dal valore *TRUE*. Eseguito il servizio, è possibile ricevere il seguente messaggio di conferma:

```
<message message="TRUE" messageType="SUCCESS"
receiverURL="http://www.questowebservice.it/servizio"
receiverId="0" senderId="1"
senderURL="http://www.altrowebservice.it/servizio" />
```

Questo messaggio dà la conferma al mittente che il precedente messaggio è stato eseguito con successo e che la lampada è ora accesa. Qui di seguito la formalizzazione della grammatica XML (XML Schema) per generare un **domoMessage** corretto (well-formed).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="quali
  <xs:element name="message">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="input"/>
      </xs:sequence>
      <xs:attribute name="message" use="required" type="xs:string"/>
      <xs:attribute name="messageType" use="required" type="xs:string"/>
      <xs:attribute name="receiverId" use="required" type="xs:string"/>
      <xs:attribute name="receiverURL" use="required" type="xs:string"/>
      <xs:attribute name="senderId" use="required" type="xs:string"/>
      <xs:attribute name="senderURL" use="required" type="xs:string"/>
    </xs:complexType>
```

```
  </xs:element>
  <xs:element name="input">
    <xs:complexType>
      <xs:attribute name="name" use="required" type="xs:string"/>
      <xs:attribute name="type" use="required" type="xs:string"/>
      <xs:attribute name="value" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

2 Conclusioni

Il linguaggio domoML, in conclusione, ha l'obiettivo e l'ambizione di diventare una lingua comune per la Domotica.

Tutti gli standard domotici esistenti potranno parlarlo, essendo capace di astrarre qualsiasi informazione riguardante un dispositivo od un servizio appartenente ad un middleware domotico, fornendo inoltre la possibilità di associare dispostivi e servizi.

La scelta di utilizzare una grammatica XML per definirlo, è sicuramente rispondente ai presupposti di generalità e libertà di scelta per quanto riguarda l'architettura o l'applicativo che desideri impiegare domoML come linguaggio per l'interazione con altri applicativi domotici.

Infatti il suo protocollo basato su stringhe permette di essere reimplementato sul runtime di qualsiasi linguaggio supporti uno *stream* di caratteri.

Il fatto poi di essere stato inserito all'interno del framework domoNET, non lo vincola comunque ne all'applicativo ne al framework stesso, ma può essere adottato da chiunque voglia sfruttato le capacità di astrazione della informazioni del linguaggio stesso.

Riferimenti bibliografici

[1] Cay Horstmann, *Fondamenti di Java 2,* (Second Edition 2000 McGraw Hill).

[2] Elliotte Rusty Harold, *Java Network Programming,* (Second Edition 2000 O'Reilly).

[3] Cay Horstmann, Gary Cornell, *Java 2, Tecniche avanzate,* (Second Edition 2000 McGraw Hill).

[4] J. F. Kurose , Keith W. Ross, *Computer Networking,* (Second Edition 2003 Addison Wesley).

[5] Micheal J. Young, *XML Passo per Passo,* (Second Edition 2001 Microsoft Press).

[6] Steven Holzer, *XML Tutto e Oltre,* (Apogeo 2001).

[7] Bianchi Bandinelli R., *Lucidi del corso di Domotica (CLS Informatica, a.a. 2005/06)* http://www.di.unipi.it/~bandinel/domotica.html

[8] Konnex Association, *http://www.konnex.org*

[9] Konnex Standard System Specification: *Architecture Vol.3 Part I (Konnex Association, 2001)* (non di dominio pubblico)

[10] UPnP Forum, *http://www.upnp.org*

[11] UPnP Device Architecture 1.0 (UPnP Forum, 2003) *http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf*

[12] Mahmoud Q.H., *Registration and Discovery of Web Services Using JAXR* (2002)

[13] Topley K., *Java Web Services in a nutshell* (O'Reilly,2003)

[14] *Java Web Services Tutorial* (Sun Microsystems, 2004)

[15] McLaughlin B., *Java and XML 2nd Edition: Solutions to Real-World Problems* (O'Reilly, 2001)

[16] Harold E.R., *XML Bible 2nd Edition* (John Wiley and Sons, 2001)

[17] Hyde P., *Java Thread Programming: The Authoritative Solution* (SAMS, 1999)

[18] Horstmann C.S., Cornell G., *Core Java 2 Volume II:Advanced Features* (Prentice Hall, 1999)

[19] *http://www.netbeans.org/kb/articles/form getstart40.html*

[20] *http://java.sun.com/developer/technicalArticles/WebServices/jaxrws/*

[21] *http://java.sun.com/webservices/tutorial.html*

[22] *http://java.sun.com/docs/books/tutorial/native1.1*

[23] Nichols, B.A. Myers, M. Higgins, J. Hughes, T.K. Harris, R. Rosenfeld and M. Pignol, *Generating Remote Control Interfaces for Complex Appliances*, Proceedings of the 15th annual ACM symposium on user interface software and technology, ACM Press, October 2002.

[24] F. Furfari, C. Soria, V. Pirrelli, O. Signore, R. Bandinelli, *NICHE: Natural Interaction in Computerized Home Environment*, Ercim News no. 58, July 2004.

[25] L. Tarrini, V. Miori, *LIGHT: XML-Innovative Generation for Home Networking Technologies*, Ercim News no. 62, July 2005.

[26] V. Miori, L. Tarrini, M.Manca, and G. Tolomei, *An Open Standard Solution for Domotic Interoperability*, IEEE TRANSACTIONS on Consumer Electronics, Vol.52, num. 1, February 2006

[27] V. Miori, L. Tarrini, M.Manca, and G. Tolomei, *An innovative, open-standards solution for Konnex interoperability with other domotic middlewares,* Konnex scientific conference 2005, Pisa, Italy. Atti pubblicati su CD-Rom.

[28] Vittorio Miori, *La casa intelligente-Ricerca scientifica e futuri scenari*, ScienzaOnline, numero 21, anno 2

[29] V. Miori, L. Tarrini, M.Manca, and G. Tolomei, *DomoNet: A framework and a prototype for interoperability of domotics middlewares based on XML and WebServices*. 2006 IEEE International Conference on Consumer Electronics (ICCE) (Las Vegas, USA, 7-11 gennaio 2006).

[30] Dario Russo, *La domotica e Internet. Una soluzione per l'interoperabilità*. Tesi di Laurea in Informatica. 2006 Università di Pisa.