Una soluzione prototipale domotica per il Progetto C.A.S.A.

Vittorio Miori, Dario Russo

Indice

1 Introduzione

Il progetto C.A.S.A. in collaborazione tra la Società AlmaViva e ISTI/CNR, è rivolto alla realizzazione di un demo-center capace di mostrare le innovazioni tecnologiche che si possono introdurre all'interno di una abitazione al fine di renderla più sicura e confortevole.

La realizzazione ha proposto alcune soluzioni pensate esplicitamente per persone anziane e disabili, per il controllo remoto dei dispositivi installati sia attraverso una interfaccia web che tramite palmare dotato di un sintetizzatore e riconoscitore vocale. Per la realizzazione delle funzionalità è stata richiesta l'integrazione di più tecnologie domotiche tra loro altrimenti incompatibili.

2 DomoNet

L'oggetto della ricerca è stata la realizzazione di un framework per l'interoperabilità tra standard domotici differenti e per natura incompatibili in quanto aventi caratteristiche diverse, ed il controllo remoto di essi attraverso un'interfaccia unica e ciò indipendentemente dalla loro tecnologia di appartenenza.

DomoNet è un software open source rilasciato sotto licenza GPL che proporre una soluzione diversa dagli approcci fino ad ora documentati in letteratura, al problema della non interoperabilità tra gli standard domotici attualmente presenti sul mercato ed il loro controllo remoto attraverso un'interfaccia universale indipendente dalla tecnologia di appartenenza.

2.1 Lo stato dell'arte e l'innovazione

Le soluzioni proposte a progetti affini, si basano essenzialmente su due caratteristiche fondamentali: l'uso di una sorta di *driver* per categoria di dispositivo domotico e la replicazione virtuale del dispositivo da integrare nel sistema, in ogni standard domotico.

L'uso di un *driver* per ogni categoria di dispositivo domotico permette al motore del sistema di pilotare e controllare correttamente ed in maniera più automatizzata le categorie di dispositivi implementati. Questo significa che, per ogni categoria di dispositivo, occorre implementare uno strato di software. Se non è stato implementato il *driver* per una determinata categoria, si rischia di non usare il dispositivo o, nelle migliori delle ipotesi, di sotto-usarlo. Per quel che riguarda la virtualizzazione, ci permette di vedere i dispositivi domotici anche all'interno degli standard domotici diversi da quello reale di appartenenza, simulandolo nel sistema. Se ho n dispositivi ed m reti domotiche, vuol dire che logicamente, nel sistema, ho $n * m$ dispositivi dei quali $n * m - n$ sono copie (virtualizzati). Questo implica una complicazione nella gestione dei dispositivi (consistenza dei dati, ad esempio) e un uso di risorse di calcolo alto con n ed m significativamente alti.

La soluzione proposta si basa invece sull'uso di un *driver*, cioè un *gateway* (chiamato *techManager*), per tecnologia domotica. Concentrandosi sulla tecnologia, qualunque sia il dispositivo connesso alla rete domotica, l'applicativo realizzato sarà grado di pilotarlo e controllarlo senza dover scrivere altro software. Inoltre, scegliendo di astrarre il concetto di dispositivi e renderli visibili tra loro in una super-rete, non occorrerà ricorrere alla loro virtualizzazione (illustrazione 1).

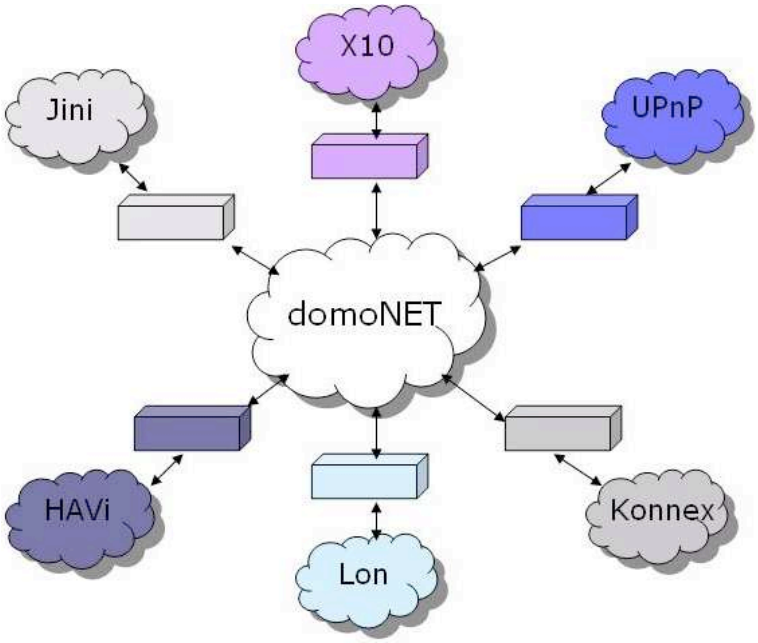

Illustrazione 1: Architettura generale di DomoNet

2.2 Architettura in generale

Questo approccio si rileva particolarmente utile nel progetto C.A.S.A. AlmaViva - CNR. Infatti, vista la grossa quantità di dispositivi diversi presenti nell'installazione del demo-center, è bastato semplicemente sviluppare i *gateway* per le tecnologie introdotte nel demo-center (*Konnex*, *UPnP*, *Bticino*). L'astrazione del concetto di dispositivo, inoltre, ha permesso di implementare facilmente il controllo remoto attraverso un'unica interfaccia software in quanto, grazie al livello di astrazione implementato dei dispositivi installati fisicamente, le differenze tecnologiche vengono appiattite. La scelta di interfacciare *DomoNet* con il mondo esterno attraverso l'uso di tecnologie standard ed aperte come i *web services* ed *XML*, ha permesso uno sviluppo rapido e sicuro delle applicazioni di controllo tramite intefaccia web e palmare.

Il livello astratto è implementato attraverso un linguaggio basato su *XML* chiamato *domoML*. Con questo linguaggio è possibile descrivere tutti i dispositivi domotici (chiamati *domoDevice*) ed i relativi messaggi ed interazioni (chiamati *domoMessage*). *DomoML* è un linguaggio intermedio da e verso il quale tradurre descrizioni e azioni.

Per implementare la cooperazione, *DomoNet* è in grado di catturare un messaggio proveniente dal *techManager* di una tecnologia, convertirlo in *domoMessage* per poi indirizzarlo al modulo gestore della tecnologia di destinazione che provvederà a riconvertirlo in modo che possa essere eseguito (illustrazione 2).

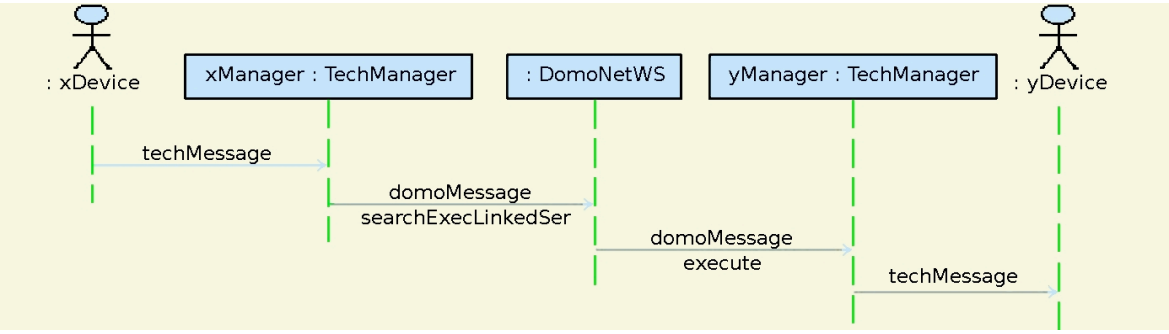

Illustrazione 2: Esempio di interoperabilità

2.3 DomoML

La ricerca si è focalizzata nella definizione di un linguaggio universale per implementare il livello di astrazione su cui si basa il funzionamento di *DomoNet*.

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 (*tech manager*), 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.

2.3.1 DomoDevice

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 in formazioni come:
    - *description*: una descrizione in lingua naturale del domoDevice;
    - *id*: valore usato per prendere e generare il *domoDeviceId*. Identifica il *domoDevice* all'interno del sistema;
    - *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;
- *URL*: usato per prendere e generare il *domoDeviceId*. Identifica l'istanza di *DomoNet* e che tiene di dispositivo.

Tutti i campi sono opzionali eccetto per l'*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 *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 naturale 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 serivizi 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 del lavoro di ricerca** in quanto permette di risolvere il problema della cooperazione tra dispositivi reali eterogenei. Questo tag è opzionale. I suoi attributi sono:
    - *URL*: l'identificativo dell'istanza di *DomoNet* che gestisce il *domoDevice* da collegare al servizio;
    - *id*: l'identificativo del *domoDevice* da collegare al servizio all'interno dell'istanza di *DomoNet*;
    - *service*: il nome del servizio del *domoDevice* individuato dai precedenti attributi da collegare;

- *ifInput:* attributo opzionale per porre una condizione di esecuzione dell'interoperabilità. Identifica il nome di un *input* del servizio nel quale è dichiarato;
- *hasValue*: attributo legato ad *ifInput*. Se l'input dichiarato in *ifInput* ha il valore dichiarato in *hasValue*, allora viene eseguita l'interoperabilità e quindi il tag *linkedService.*

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;
  - *value*: se non espresso il campo *from*, assegna all'input dichiarato in *to* il valore costante dichiarato con questo attributo.

Un esempio di *domoDevice* per una semplice lampadina di tecnologia *Konnex* che alla sua accensione manda in *play* un brano di un *media-renderer UPnP* (illustrazione 3):

```
<device description="Bathroom light" id="7"
  url="http://myhost/axis/domoNetWS" manufacturer="Pholips"
  positionDescription="Bathroom" serialNumber="1.1.6"
  tech="KNX" type="light bulb" category="DomoLight"
  sub-category="DomoDimmer">
   <service description="Get state of the light"
    name="getPower"
    output="BOOLEAN" prettyName="Light status" />
   <service description="Set state of the light" name="setPower"
    prettyName="Set status">
     <input description="" name="value" type="BOOLEAN">
        <allowed value="0" />
        <allowed value="1"/>
     </input>
     <linkedService id="42" url="" service="Play"
       ifInput="value" hasValue="1">
        <linkedInput to="mediaRendererId" value="52" />
        <linkedInput to="mediaRendererURL" value="" />
        <linkedInput to="mediaContainerId" value="0/Music" />
        <linkedInput to="mediaContentId"
          value="Music/NESSUNDORMA.MP3" />
     </linkedService>
   </service>
   <service description="Set the intensity of the light"
    name="setIntensity" prettyName="Set the intensity">
     <input description="The intensity" name="value"
      type="INT" />
   </service>
   <service description="Get the intensity of the light"
    name="getIntensity" prettyName="Get the intensity"
    output="INT" />
   </service>
</device>
```

Illustrazione 3: Esempio di domoDevice

2.3.2 DomoMessage

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;
     - *UPDATE*: se è stato ricevuto un messaggio da un *tech-manager* per aggiornare lo stato di un dispositivo;

- *receiverId*: l'identificatore del *domoDevice* ricevente del messaggio;
- *receiverURL*: l'url dell'istanza di *DomoNet* che contiene il *domoDevice* richiesto;
- senderId: l'identificatore del *domoDevice* mittente del messaggio;
- *senderURL*: l'url dell'istanza di *DomoNet* 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 è (illustrazione 4):

```
<message message="setPower"
  messageType="EVENT"
  receiverId="7"
  receiverURL="http://myhost/axis/domoNetWS"
  senderId="7"
  senderURL="http://myhost/axis/domoNetWS">
    <input name="value" type="BOOLEAN"
      value="1" />
</message>
```

Illustrazione 4: Esempio di domoMessage

2.4 I principali techManager implementati

2.4.1 Konnex

Il KNXManager è il modulo che gestisce la tecnologia Konnex e si basa sulle librerie open source *Calimero* (http://calimero.sourceforge.net).

Il costruttore del modulo prende come parametro la *url* e il numero di porta del servizio che è connesso ai dispositivi.

Al tempo di *startup* esegue la connessione al server e l'inizializzazione delle strutture che si occupano della conversione dei *datatype*, esattamente da *domoML.DomoDevice.domoDevice.DataType* agli identificatori Major e Minor, usati per convertire il tag input del *domoMessage* in un valore fruibile per la tecnologia.

Dopo le inizializzazioni il modulo carica i dispositivi disponibili parsando un file *XML* generato dall'applicazione *ETS 3* (*EIB Tools Software* è un software distribuito dalla *Konnex Association* per configurare dispositivi).

Il file di configurazione contiene le descrizioni e tutte le funzionalità che il sistema Konnex può offrire. Come esempio, di seguito è mostrato un pezzo di file di configurazione per il servizio di un

dispositivo:

```
<row>
  <colValue nr="1">0/0/1 Comando Uscita A</colValue>
 <colValue nr="2">
   0: Comando,tasto sinistro superiore-Commutazione
 </colValue>
 <colValue nr="3">
   1.1.4 16196.. Pulsante 4 canali
 </colValue>
 <colValue nr="4">S</colValue>
 <colValue nr="5">-</colValue>
 <colValue nr="6">C</colValue>
 <colValue nr="7">-</colValue>
 <colValue nr="8">W</colValue>
 <colValue nr="9">T</colValue>
 <colValue nr="10">U</colValue>
 <colValue nr="11">
   16196.. Pulsante 4 canali
 </colValue>
 <colValue nr="12">
   16196 On-Off-Dimmer-Tapparelle-Led
 </colValue>
 <colValue nr="13">1 bit</colValue>
 <colValue nr="14">Basso</colValue>
 <colValue nr="15">0/0/1</colValue>
</row>
```

Da questo pezzo, e da ogni tag *row* presente nel file, è possibile individuare:

- nel secondo colValue: il nome del servizio;
- nel terzo colValue (la parte iniziale): il real address del dispositivo;
- nel settimo colValue: se il campo è leggibile (settato a R);
- nell'ottavo colValue: se il campo è scrivibile (settato a W);
- nel nono colValue: se il campo è trasmissibile (settato a T);
- nell'undicesimo colValue: il nome del device che offre il servizio;
- nel dodicesimo colValue: la descrizione del servizio;
- nel quindicesimo colValue: l'indirizzo di gruppo associato al servizio;

Gli altri colValue non sono significativi allo scopo proposto dal *KNXManager*. Interessano i servizi che sono trasmissibili e almeno o leggibili o scrivibili. Un dispositivo viene descritto in tutti i suoi servizi unendo insieme tutti gli undicesimi colValue con la stessa etichetta.

Questo esempio produce il seguente domoDevice:

```
<device description="" id="" manufacturer=""
 postionDescription="" serialNumber="" tech="KNX"
 type="16196.. Pulsante 4 canali" url="">
    <service description="Get the status"
      output="BOOLEAN" outputDescription="The value"
      prettyName="Get status" name="GET_STATUS" />
    <service name="0/0/1"
      description="16196 On-Off-Dimmer-Tapparelle-Led"
      prettyName="16196 On-Off-Dimmer-Tapparelle-Led">
        <input description="The value" name="value"
          type="BOOLEAN">
            <allowed value="TRUE" />
            <allowed value="FALSE" />
        </input>
    </service>
</device>
```

Come catturare i tech message dal bus Konnex

Tutti i messaggi che transitano sul bus Konnex vengono catturati dal KNXManager attraverso l'uso di un listener in ascolto sul bus. E interessante monitorare il bus per due scopi:

1. in attesa di risposta derivata dall'esecuzione di un *domoMessage*: in questo caso è stato implementato un sistema a semafori e dove i produttori sono i dispositivi che scrivono sul bus e il *tech-manager* è il consumatore. Il real address del mittente di ogni messaggio che transita sul bus viene confrontato con quello di cui siamo in attesa. Se l'indirizzo viene riconosciuto in un limite di tempo prefissato viene generato con i dati del *tech-message* il domoMessage di risposta altrimenti viene generato un *domoMessage* di fallimento. Un esempio tipico di questo meccanismo è l'interrogazione di un dispositivo per conoscerne lo stato;
2. per implementare la cooperazione: ogni messaggio che transita sul bus viene catturato dal *listener*. Il *tech-message* estrapola il *real address* del dispositivo mittente e riconosce il servizio tramite l'indirizzo di gruppo a cui è rivolto il comando. Con questi dati, è possibile ricavare il *domoDevice* e la descrizione del servizio. Il sistema può quindi verificare se è impostato il tag *linkedService* ed agire di conseguenza.

2.4.2 UPnP

L'*UPnPManager* è il modulo che gestiche la tecnologia *UPnP* e si basa sulla libreria *Cyberlink* (http//sourceforge.net/projects/cgupnpjava).

A tempo di startup, il l'*UPnPManager* inizializza il *ManagerPoint*: il listener, cuore del manager, che permette di aggiungere, rimuovere, testare l'*hartbeat* e amministrare i pacchetti dei dispositivi. Il *ManagerPoint* provvede ad inizializzare le strutture dati per la conversione da *domoML.DomoDevice.domoDevice.DataType* ai simboli usati per questa tecnologia e viceversa.

Quando viene aggiunto un dispositivo alla rete *UPnP*, questo si annuncia dando la propria descrizione e quella dei servizi che può offrire. Queste informazioni sono memorizzate in diversi file *XML* che la libreria provvede a parsare e a darne l'albero corrispondente. A questo punto diventa facile ottenere le informazioni cercate. Un esempio di descrizione del dispositivo è:

```
<device>
  <deviceType>urn:schemas-upnp-org:device:light:1
  </deviceType>
  <friendlyName>CyberGarage Light Device</friendlyName>
  <manufacturer>CyberGarage</manufacturer>
  <manufacturerURL>http://www.cybergarage.org
  </manufacturerURL>
  <modelDescription>CyberUPnP Light Device
  </modelDescription>
  <modelName>Light</modelName>
  <modelNumber>1.0</modelNumber>
  <modelURL>http://www.cybergarage.org</modelURL>
  <serialNumber>1234567890</serialNumber>
  <UDN>uuid:cybergarageLightDevice</UDN>
  <UPC>123456789012</UPC>
  <iconList>
   …
  </iconList>
  <serviceList>
   …
  </serviceList>
  <presentationURL>http://www.cybergarage.org
 </presentationURL>
</device>
```

Da questo pezzo di codice si trova:

- dal tag *friendlyName*: la descrizione del dispositivo;
- dal tag *deviceType*: il real address del dispositivo;
- dal tag *modelName*: il tipo del dispositivo;
- dal tag *serialNumber*: il serial number.

Quello dei servizi offerti (chiamati *actions*) è:

```
<actionList>
 <action>
  <name>SetPower</name>
  <argumentList>
   <argument>
```

```
    <name>Power</name>
    <relatedStateVariable>
        Power
    </relatedStateVariable>
    <direction>in</direction>
   </argument>
   <argument>
    <name>Result</name>
    <relatedStateVariable>
       Result
    </relatedStateVariable>
    <direction>out</direction>
   </argument>
  </argumentList>
 </action>
 <action>
  <name>GetPower</name>
  <argumentList>
   <argument>
    <name>Power</name>
    <relatedStateVariable>Power</relatedStateVariable>
    <direction>out</direction>
   </argument>
  </argumentList>
 </action>
</actionList>
<serviceStateTable>
 <stateVariable sendEvents="yes">
  <name>Power</name>
  <dataType>boolean</dataType>
  <allowedValueList>
   <allowedValue>0</allowedValue>
   <allowedValue>1</allowedValue>
  </allowedValueList>
  <allowedValueRange>
   <maximum>123</maximum>
   <minimum>19</minimum>
   <step>1</step>
  </allowedValueRange>
 </stateVariable>
 <stateVariable sendEvents="no">
  <name>Result</name>
   <dataType>boolean</dataType>
```

```
 </stateVariable>
</serviceStateTable>
```

Da questo pezzo di codice si deduce:

- dal tag *action* - *name*: il nome e il pretty name del servizio;
- dal tag *argument* - *name*: il nome del parametro di input;
- dal tag *relatedStateVariable*: la chiave da cercare nella serviceStateTable per trovare le altre informazioni del parametro;
- dal tag *direction*: se è out è un parametro di output quindiè aspettato un valore di ritorno all'esecuzione del servizio; se è in è un parametro di input;
- dal tag *datatype*: il *datatype* del parametro;
- dal tag *allowedValue*: i valori permessi per quel parametro. Per ogni *action* è previsto un set di *argument*. Ogni *argument* ha un *datatype* da essere poi convertito in *domoML.DomoDevice.domoDevice.DataType*. Di ogni *argument* occorre verificare il tag *direction* per sapere se la variabile di stato deve essere usata come parametro di input o di output.

Il corrispondente *domoDevice* risultante è:

```
<device description="CyberGarage Light Device" id=""
 manufacturer=""       positionDescription=""
 serialNumber="1234567890" tech="UPNP" type="Light">
    <service description="" name="SetPower"
      output="BOOLEAN" prettyName="SetPower">
       <input description="" name="Power"
         type="BOOLEAN">
           <allowed value="0" />
           <allowed value="1" />
       </input>
    </service>
    <service description="" name="GetPower"
      output="BOOLEAN" prettyName="GetPower" />
</device>
```

Come catturare i *tech-message* dalla rete *UPnP*

L'UPnPManagerPoint implementa anche le funzioni di listener in attesa di messaggi che transitano sulla rete. Quando un nuovo messaggio transita, ne viene calcolato il codice di sottoscrizione

del servizio e il *domoDeviceId* del dispositivo mittente. Con queste informazioni, passate al web service, è possibile verificare la presenza di *linkedService* associati al servizio invocato.

2.4.3 Implementare il proprio tech-Manager

Per implementare un nuovo modulo tech manager, è necessario estendere la classe *DomoNetWS.techManager.TechManager* e creare una nuova istanza della nuova classe nella DomoNetWS.DomoNetWS.managerList usando come chiave l'identificatore della tecnologia (essa deve essere contenuta in domoML.domoDevice.DomoDevice.DomoTech). Il nuovo modulo deve implementare i metodi astratti che sono:

- public void addDevice(final DomoDevice domoDevice, String address): per aggiungere un dispositivo alla lista dei domoDevice del sistema; deve chiamare i metodi doubleHash.add() e addDomoDevice() del web service;
- public DomoMessage execute(DomoMessage domoMessage) throws Exception: per eseguire un *domoML.domoMessage.DomoMessage*; deve essere in grado di convertire il *domoMessage* in *tech-message* e vice versa;
- public void finalize(): azioni da eseguire quando viene chiusa l'applicazione.

3 Ulteriori attività di ricerca

Finalità principale delle ulteriori attività di ricerca sono state l'ampliamento delle funzionalità domotiche già rese disponibili dal demo-center AlmaViva - CNR.

Tale finalità hanno comportato lo studio di soluzioni alternative per poter aggiungere nuovi dispositivi domotici. La ricerca è stata focalizzata sullo studio ed lo sviluppo per il supporto tecnologico di questi all'interno del *framework DomoNet*. Infatti, sono state introdotte nuove tecnologie fino a quel momento non considerate dal *framework* ampliandone così il parco e le possibilità. Sono state inoltre perseguite finalità pratiche fornendo soluzioni più accurate ai problemi che coinvolgono anziani e disabili.

L'ampliamento delle funzionalità e dei dispositivi all'interno del demo center ha permesso uno studio maggiore del comportamento del *framework* in configurazioni e situazioni non ancora pienamente testate.

La ricerca è proseguita individuando i "colli di bottoglia" del sistema, affrontare le problematiche relative alla sicurezza e progettare una modifica architetturale del *framework* al fine di aumentarne le performance.

Per eliminare i colli di bottiglia ed aumentare le performance del sistema, sono state necessarie delle revisioni su parti critiche sottoposte a maggiore utilizzo. Inoltre, poche modifiche architetturali hanno permesso di ottenere un sistema più scalabile e modulare rispetto all'attuale, sfruttando maggiormente i i vantaggi di tali paradigmi.

L'attività di ricerca si è anche concentrata sullo studio e confronto con altri software similari già esistenti al fine di poter offrire un software sempre più aggiornato, competitivo e completo.

Per quel che concerne la sicurezza sono stati implementati algoritmi di criptazione ed autenticazione delle utenze che dispongono del sistema al fine di evitare accessi indesiderati.

Un'altra attività di ricerca effettuata è stata quella di rendere disponibile il framework anche su piccoli *device embedded*, solitamente con risorse hardware ridotte e di particolare interesse nel mondo della domotica, al fine di poter installare fisicamente il software su dispositivi che rimangono poi nascosti all'interno delle mura domestiche e quindi di non funzionare necessariamente solo su

personal computer completi.

4 Conclusioni

Il software è stato ampiamente testato ed è attualmente in uso nel demo-center del *domotics lab* presso l'ISTI del CNR di Pisa.

Il SW OS "*DomoNet Architecture*" è attualmente:

– Utilizzato dal Politecnico di Milano in più abitazioni dimostrative per persone disabili

– Oggetto di ulteriore sviluppo da parte del Politecnico di Torino e del Politecnico di Milano

– Oggetto di un apposito tirocinio ad Ingegneria Informatica presso il Politecnico di Milano

– Commentato perlomeno nei seguenti articoli di diffusione internazionale:

- Miori V., Russo, D., Aliberti, M. - Domotic Technologies Incompatibility Becames User Transparent - ACM Communication, Accettato ed in attesa di pubblicazione
- Miori V., Russo, D., Aliberti, M. - An Informatics Research Contribution to the Domotic Take-Off - ERCIM January 2008 n. 72, pp. 54 – 55.
- Miori V., Tarrini L., Manca M., Tolomei G. - An open standard solution for domotic interoperability. In: IEEE Transactions on Consumer Electronics, vol. 52 (1) pp. 97-103. IEEE, 2006, ISI/JCR.
- Miori V., Tarrini L., Manca M., Tolomei G. DomoNet: a framework and a prototype for interoperability of domotics middlewares based on XML and WebServices. In: ICCE - International Conference on Consumer Electronics (Las Vegas, 7-11 January 2006). Proceedings, pp. 117 - 118. Suvisoft Oy Ltd. Hermiankatu 3A FIN-33720 Tampere FINLAND (ed.). IEEE, 2006, ISI/JCR.
- Hyung-Jun Yim, Il-Jin Oh, Yun-Young Hwang, Kyu-Chul Lee, Kangchan Lee, and Seungyun Lee, "Design of DPWS Adaptor for Interoperability between Web Services and DPWS in Web Services on Universal Networks," Convergence Information Technology, International Conference on, pp. 1032-1039, 2007 International Conference on Convergence Information Technology (ICCIT 2007), 2007.
- Mori, G., Paternò, F., and Spano, L. D. 2008. Exploiting Web Services and Model-Based User Interfaces for Multi-device Access to Home Applications. In interactive Systems. Design, Specification, and Verification: 15th international Workshop, DSV-IS 2008 Kingston, Canada, July 16-18, 2008 Revised Papers, T. C. Graham and P. Palanque, Eds. Lecture Notes In Computer Science, vol. 5136. Springer-Verlag, Berlin, Heidelberg, 181-193. DOI= http://dx.doi.org/10.1007/978-3-540-70569-7_18
- Yamabe, T.; Takahashi, K., "UWS Broker for Ubiquitous Web Services Dynamic Discovery," Intelligent Pervasive Computing, 2007. IPC. The 2007 International Conference on , vol., no., pp.204-209, 11-13 Oct. 2007 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4438425&isnumber=4438371
- Yun-Young Hwang, Hyung-Jun Yim, Il-Jin Oh, Kyu-Chul Lee, "The Universal Adaptor for Dynamic Ubiquitous Services Discovery and Binding," Computer Science and Software

Engineering, International Conference on, vol. 3, pp. 515-518, 2008 International Conference on Computer Science and Software Engineering, 2008.

- Jessica Dìaz, Agustìn Yague, Pedro P. Alarcòn, Juan Garbajosa, "A Generic Gateway for Testing Heterogeneous Components in Acceptance Testing Tools," Commercial-off-the-Shelf (COTS)-Based Software Systems, International Conference on, pp. 110-119, Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008), 2008.
- Yun-Young Hwang, Il-Jin Oh, Hyung-Jun Im, Kyu-Chul Lee, Kangchan Lee, Seungyun Lee, "Universal Service Discovery Protocol", Convergence Information Technology, International Conference on, pp. 2380-2385, 2007 International Conference on Convergence Information Technology (ICCIT 2007), 2007.
- Mauricio Esteban Pardo, "A domotic system with remote access based on Web services.", Journal of Computer Science & Technology (Magazine/Journal), Volume: 8 Issue: 2 Page: 91(6), Graduate Network of Argentine Universities with Computer Science Schools (RedUNCI), July 1, 2008
- Ming-Chien Yang; Norman Sheng; Brandon Huang; Jethro Tu, "Collaboration of Set-Top Box and Residential Gateway Platforms," Consumer Electronics, IEEE Transactions on , vol.53, no.3, pp.905-910, Aug. 2007
- Perumal, T.; Ramli, A.R.; Chui Leong, "Design and implementation of SOAP-based residential management for smart home systems," Consumer Electronics, IEEE Transactions on , vol.54, no.2, pp.453-459, May 2008
- MC Yang, N Sheng, B Huang, J Tu "Interoperability between set-top box and residential gateway platforms", Communications and Information Technologies, 2007. ISCIT '07. International Symposium on, 2007, pp. 1286-1290, Oct. 2007
- Fabio Paternò, Carmen Santoro, Lucio Davide Spano, "Designing Usable Applications based on Web Services", Proceedings of the First Workshop on the Interplay between Usability Evaluation and Software Development, Pisa, Italy, September 24, 2008
- Witte, L.; Telkamp, G., "Domotics and Infomobility in the ASK-IT Project", ASK-IT 1st International Conference, Nizza (Frankreich), 2006
- Torbensen, R., "OHAS: Open home automation system," Consumer Electronics, 2008. ISCE 2008. IEEE International Symposium on , vol., no., pp.1-4, 14-16 April 2008

5 Bibliografia

[1] E. Niemelä and T. Vaskivuo, "Agile Middleware of Pervasive Computing Environments", Proceedings of IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 192-197, March 2004

[2] Lorente, S. "Key issues regarding domotic applications", Proceedings of the International Conference on Information and Communication Technologies: From Theory to Applications, 2004., pp. 121-122, April 2004

[3] Scholten, H., van Dijk, H., de Cock, D., Preneel, B., D'Hooge, M., Kung, A., "Secure Service Discovery in Home Networks", Proceedings of ICCE '06 - IEEE International Conference on Consumer Electronics, pp. 115- 116, Jan 7-11, 2006.

[4] Tokunaga, E., Ishikawa, H., Kurahashi, M., Morimoto, Y., and Nakajima, T. 2002. “A Framework for Connecting Home Computing Middleware”. Proceedings of the 22nd international Conference on Distributed Computing Systems, ICDCSW, IEEE Computer Society, Washington, DC, pp. 765-770, July 02-05, 2002

[5] Pellegrino, P., Bonino, D., and Corno, F. 2006. “Domotic house gateway”. Proceedings of the 2006 ACM Symposium on Applied Computing, SAC '06. ACM Press, New York, NY, pp. 1915-1920, Dijon, France, April 23 - 27, 2006.

[6] Nichols, J., Myers, B.A., Higgins, M., Hughes, J., Harris, T.K., Rosenfeld, R. and Pignol, M. “Generating remote control interfaces for complex appliances”. Proceedings of the 15th annual ACM symposium on User interface software and technology, ACM Press., pp. 161—170, 2002.

[7] Miori V., Tarrini L., Manca M., Tolomei G. “An open standard solution for domotic interoperability”. In: IEEE Transactions on Consumer Electronics, vol. 52 (1) pp. 97 - 103. IEEE, 2006.

[8] Papazoglou, M.P. and Georgakopolous, D. “Service Oriented Computing”. In Communications of the ACM, Vol. 46 (10), pp. 24-28, Oct. 2003.

[9] Furfari F., Soria C., Pirrelli V., Signore O., Bianchi Bandinelli R. NICHE: “Natural Interaction in Computerised Home Environments”. In: ERCIM News, vol. 58 pp. 55-56. Special issue: Automated Software Engineering. ERCIM EEIG, 2004.

[10] Darking, M., Whitley, E. A., and Dini, P. “Governing diversity in the digital ecosystem”. Communications of the ACM, Vol. 51 (10), pp.137-140, Oct. 2008.