**(Versione documento: draft - 1.1)**

# Definizione architettura del middleware SHELL: considerazioni sulle modalità di interazione tra gli HOST

**Dario Russo, Vittorio Miori**

Progetto SHELL - Ecosistemi domestici condivisi ed interoperabili per ambienti di vita sostenibili, confortevoli e sicuri.

Progetto SHELL - Ecosistemi domestici condivisi ed interoperabili per ambienti di vita sostenibili, confortevoli e sicuri.

## Indice

# 1. Sommario

L'architettura SHELL prevede uno o più sistemi reali capaci di fare da ponte per la comunicazione tra i dispositivi appartenenti agli HOST, anche se essi sono disomogenei. I suoi obiettivi sono quelli di permettere:

1. la gestione dei dispositivi presenti nell'ambiente assicurando la convivenza e l'interazione anche tra quelli che accedono alla rete con protocolli diversi;
2. un'espansione delle funzionalità realizzabili dalla rete domotica rispetto a quelle fornite dal modello tradizionale (semplice interazione sensore-attuatore).

L'HOST rappresenta un valore aggiunto che si va a sommare al tradizionale modello domotico al fine di renderlo più funzionale. In genere si tratta di un comune personal computer connesso a una o più reti di dispositivi presenti nell'ambiente con lo scopo di monitorare il loro stato (ricevendo le loro notifiche) e di inviare comandi da eseguire.
I compiti che il framework SHELL dovrà essere in grado di svolgere si possono riassumere in:

1. funzionalità necessarie per controllare diversi sistemi domotici utilizzando un unico protocollo di comunicazione di alto livello;
2. funzionalità necessarie per permettere l'automazione internetwork (interazione tra dispositivi appartenenti a sistemi domotici differenti).

# 2. Abstract

The SHELL architecture provides for one or more real systems able to act as a bridge for the communication between the devices belonging to the different HOSTs, even if they are uneven. Its goals are to enable:

1. the management of the devices inside the environment by ensuring the co-existence and the interaction even among those who access the network using different protocols;
2. the increase of the capabilities achievable by the home automation network in comparison to those provided by the traditional model (simple sensor-actuator interaction).

The HOST is an added value to the traditional home automation model in order to make it more efficient. Generally it is a common personal computer connected in the environment to one or more networks of devices with the aim to monitor their status (receiving their notifications) and to send commands to be executed.
The tasks that the SHELL framework will have to be able to carry out can be summarized in:

1. necessary features to control different home automation systems using a single high-level communication protocol;
2. necessary functionalities allowing the automation internetwork (interaction between devices belonging to different home automation systems).

## 3. Introduzione

L'architettura della piattaforma SHELL è basata su una rete di HOST che contengono al loro interno device reali e/o virtuali (Figura 1).

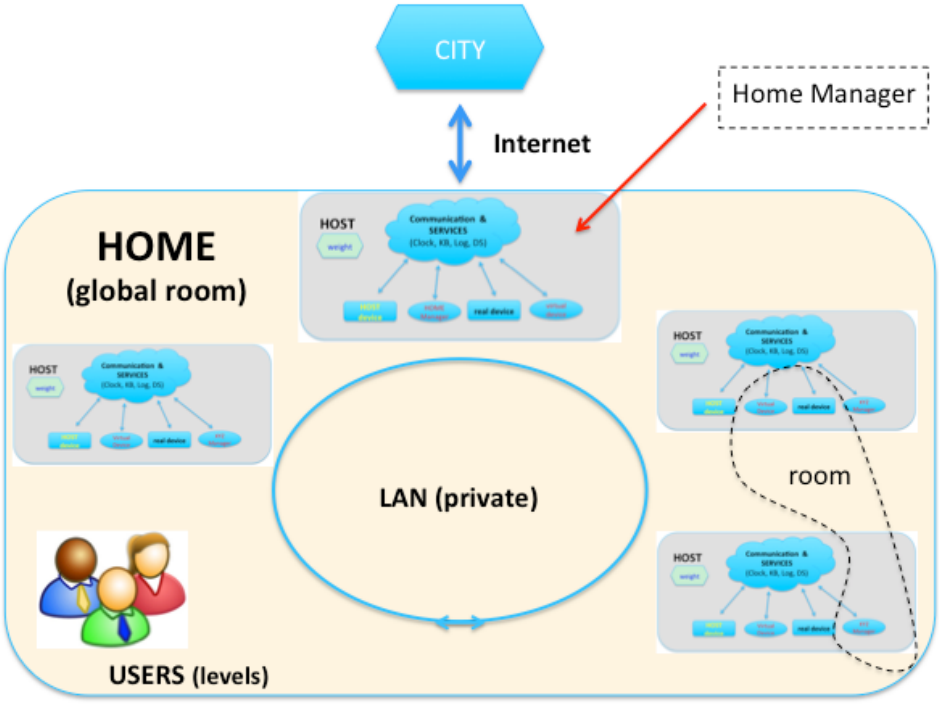



Figura 1: L'architettura SHELL

Un HOST è un sistema (embedded) reale connesso alla rete LAN che contiene l'implementazione di uno o più devices, esso quindi possiede sempre un indirizzo IP e una geolocalizzazione. I dispositivi implementati possono essere sia virtuali che reali, in quest'ultimo caso se sono nativi (SHELL compliant), l'host deve disporre dell'hardware necessario alla loro implementazione. Se invece essi sono wrapped (appartenenti a una tecnologia domotica non compatibile con il sistema SHELL), l'host deve possedere le opportune interfacce per i bus fisici richiesti, al fine della loro integrazione nel framework.
I dispositivi possono essere raggruppati in gruppi (room). Ogni gruppo ha un manager che lo gestisce e ne determina le funzionalità e lo scopo.
Ogni dispositivo può appartenere ad uno o più gruppi. Esiste sempre un gruppo predefinito chiamato home manager, che supervisiona le attività di tutta la rete di dispositivi della casa.
L'architettura SHELL prevede uno o più sistemi reali capaci di fare da ponte per la comunicazione tra i dispositivi appartenenti agli HOST, anche se essi sono disomogenei. I suoi obiettivi sono quelli di permettere:

3. la gestione dei dispositivi presenti nell'ambiente assicurando la convivenza e l'interazione anche tra quelli che accedono alla rete con protocolli diversi;
4. un'espansione delle funzionalità realizzabili dalla rete domotica rispetto a quelle fornite dal modello tradizionale (semplice interazione sensore-attuatore).

L'HOST rappresenta un valore aggiunto che si va a sommare al tradizionale modello domotico al fine di renderlo più funzionale. In genere si tratta di un comune personal

computer connesso a una o più reti di dispositivi presenti nell'ambiente con lo scopo di monitorare il loro stato (ricevendo le loro notifiche) e di inviare comandi da eseguire.
Per rendere chiaro questo concetto si consideri il caso di un interruttore KNX e di una lampada UPnP. Si vuole realizzare la seguente interazione: quando l'interruttore viene posizionato in posizione "On" la luce della lampada deve accendersi.
L'interruttore e la lampada sono implementati in mdodalità "wrapped", rispettivamente all'interno di due HOST diversi.
Una possibile soluzione consiste nell'inserire una regola di controllo all'interno dell'HOST, da eseguire solo se si verificano certe condizioni. La ricezione di una notifica "On" dall'interruttore determinerà la generazione e la trasmissione alla lampada di un comando per accendere la luce. Gli HOST svolgono quindi la funzione di traduzione: il primo riceve una notifica KNX, la interpreta e genera un messaggio indirizzato all'HOST che contiene la lampada UPnP in modalità "wrapped", che lo traduce e lo invia sul bus UPnP.
Il linguaggio utilizzato per descrivere i sistemi domotici è ShellML, un metamodello espresso nel formalismo XML, il cui compito è formalizzare tutte le caratteristiche dei devices domotici. ShellML fornisce la decrizione formale di ogni elemento del sistema domotico e in particolare la rappresentazione dei device.
L'ambiente domotico gestito dal framework SHELL è quindi composto da uno o più HOST che permettono di implementare politiche di interoperabilità e comportamenti intelligenti.
Un sistema domotico contiene, tipicamente, diversi dispositivi domotici (attuatori per porte e finestre, luci, sensori, ecc.) e un network gateway che, agendo da tunnel (inserendo quindi un'intestazione di rete), permette la trasmissione dei messaggi specifici del protocollo di basso livello, utilizzato dai dispositivi appartenenti al sistema domotico per la comunicazione, su di una tecnologia di interconnessione maggiormente versatile (es. Ethernet).
I network gateway rappresentano quindi un punto di accesso per i sistemi domotici, ma non sono programmabili e non introducono funzionalità aggiuntive all'interno del framework.
Gli elettrodomestici possono essere "stupidi", ovvero possono essere controllati solo attivando o disattivando le prese elettriche alle quali sono collegati, o"smart", cioè capaci di offrire funzionalità complesse e di controllare altri dispositivi attraverso uno specifico protocollo di comunicazione.
La presenza nello stesso ambiente di molti sistemi domotici introduce delle problematiche di interoperabilità: dispositivi appartenenti a sistemi diversi non possono interagire per via delle diverse tecnologie e dei diversi protocolli utilizzati per comunicare. Il sistema SHELL è progettato con lo scopo di risolvere questa problematica: agisce da ponte tra le diverse tecnologie e supporta complesse interazioni tra i dispositivi attraverso la comunicazione tra HOST. I compiti che il framework SHELL deve essere in grado di svolgere si possono riassumere in:

3. funzionalità necessarie per controllare diversi sistemi domotici utilizzando un unico protocollo di comunicazione di alto livello;
4. funzionalità necessarie per permettere l'automazione internetwork (interazione tra dispositivi appartenenti a sistemi domotici differenti);

# 4. Il linguaggio ShellML

**ShellML** è un linguaggio descrittivo dei dispositivi SHELL e consente di definire:

- le capacità e le funzionalità dei dispositivi;
- le specifiche tecnologiche necessarie per interfacciarsi con i dispositivi;
- le possibili configurazioni che i dispositivi possono assumere.

Per la sua rappresentazione, ShellML si appoggia al modello ontologico DogOnt.
Esso è composto di due parti: un'ontologia (DogOnt Ontology), espressa in OWL, il cui compito è formalizzare tutti gli aspetti dell'ambiente domotico, e un insieme di regole (DogOnt Rules), che facilitano il processo di modellazione generando automaticamente gli stati e le funzionalità relative ai dispositivi, e associandoli alle opportune istanze attraverso regole semantiche.
DOG (Domotic OSGi Gateway) è una piattaforma che permette l'interfacciamento, la gestione e l'integrazione di dispositivi domotici prodotti da diversi costruttori.
L'utilizzo del gateway DOG in un ambiente domotico rappresenta una possibile soluzione per permettere l'interoperabilità tra dispositivi domotici diversi e l'aggiunta di altre funzionalità. L'utilizzo di DOG a tale scopo però introduce delle problematiche di configurazione non banali.
Per il suo funzionamento, infatti, DOG si appoggia al modello DogOnt che descrive l'ambiente. L'istanza dell'ontologia, quindi, è specifica del contesto: per ogni impianto domotico in cui viene installato DOG occorre creare il file di ontologia che lo descrive.
Il problema di questa rappresentazione è che produrre un documento istanza di DogOnt è molto complesso: il formato OWL richiede infatti di organizzare le informazioni in una maniera poco intuitiva, utilizzando costrutti e ridondanze tediose.
Il dover descrivere un ambiente domotico direttamente in tale formato (attraverso l'uso di software come *Protégé*) crea molte difficoltà al progettista: richiede di essere fortemente specializzati ed è dispendioso di tempo.
Questo approccio frena quindi lo sviluppo e l'installazione di ambienti domotici basati su DOG in quanto è molto difficile produrre un file di configurazione corretto ed è comunque pensabile solo per la descrizione di sistemi semplici, dove gli elementi e le interazioni presenti siano poche.
Questo è quindi uno dei principali motivi per cui l'approccio SHELL è quello di utilizzare DogOnt come formato di descrizione generale di riferimento, dal quale ricavare è una formalizzazione che permette di ottenere una rappresentazione di ShellML: ciò viene fatto utilizzando un apposito sistema inferenziale che, interrogando l'ontologia, consente l'estrazione automatica delle informazioni necessarie alla rappresentazione di ShellML.
L'utilizzo nell'implementazione di SHELL della terminologia e delle assunzioni di DogOnt, oltre ad allineare SHELL allo stato dell'arte della letteratura di riferimento, pone le basi per facilitare lo sviluppo di applicazioni semantiche e di "ambient intelligence". DogOnt potrebbe infatti essere utilizzato come parte integrante del livello semantico del framework SHELL.

## 4.1. Funzionalità

Esse modellano le operazioni che si possono effettuare su un dispositivo. Tutte le funzionalità defniscono i comandi necessari per modificare o per interrogare una specifica proprietà del dispositivo. Sono suddivise in diverse categorie in relazione al

loro scopo:

- le ControlFunctionality modellano la capacità di controllare un dispositivo.
- le NotificationFunctionality riguardano l'abilità di un dispositivo di segnalare un proprio cambiamento di stato.
- le QueryFunctionality permettono di interrogare un dispositivo che fornirà, quindi, informazioni sulla propria configurazione attuale.

Infine le funzionalità vengono organizzate anche in base a come modificano lo stato di un particolare dispositivo:

- le ContinuosFunctionality permettono la modifica delle caratteristiche di un dispositivo in maniera continua (ad esempio il volume di un impianto audio);
- le DiscreteFunctionality sono utilizzate per quelle proprietà che possono assumere solo valori discreti.

### 4.2. Stati

Gli stati modellano le configurazioni che un dispositivo può assumere. Come per le funzionalità, anche gli stati vengono classificati in base ai valori che possono assumere:

- i ContinuosState, che riguardano quelle proprietà che possono assumere valori continui;
- i DiscreteState, associati alle proprietà che assumono solo valori discreti;

### 4.3. Notification

Modellano tutte le notifiche che i dispositivi possono generare in seguito ad una variazione di stato.

### 4.4. Command

modellano tutti i comandi per controllare ed interrogare i dispositivi.

### 4.5. StateValue

Gli StateValue modellano la tipologia di valori che gli stati dei dispositivi possono assumere: le classi ContinuosStateValue e DiscreteStateValue descrivono rispettivamente i valori continui e discreti che gli stati possono assumere. Gli stati ContinuosState e DiscreteState sono associati alle classi ContinuosStateValue e DiscreteStateValue.

Vogliamo ad esempio rappresentare una lampada a intensità luminosa regolabile. Questo dispositivo è descritto dalla classe DimmerLamp che a sua volta deriva, nell'ordine indicato, dalle classi Lamp, Lighting, HousePlant e Controllable.
La presenza di questa scala gerarchica permette al dispositivo DimmerLamp di ereditare le funzionalità definite nelle super classi:

- dalle classi Lamp, Controllable e HousePlant vengono ereditate rispettivamente le funzionalità:
    - OnOffFunctionality (che a sua volta definisce i comandi *On* e *Off* per accendere e spegnere la lampada),
    - QueryFunctionality (per permettere l'interrogazione dello stato della

lampada) e
    - StateChangeNotificationFunctionality (notifica che segnala un cambiamento di stato della lampada).
- La classe DimmerLamp è inoltre caratterizzata dalla funzionalità LightRegulationFunctionality e dallo stato LightIntensityState:
    - LightRegulationFunctionality definisce i comandi StepUpCommand, StepDownCommand e SetCommand(value) per governare l'incremento ed il decremento dell'intensità luminosita,
    - LightIntensityState tiene traccia, in percentuale, del valore dell'intensità luminosa.

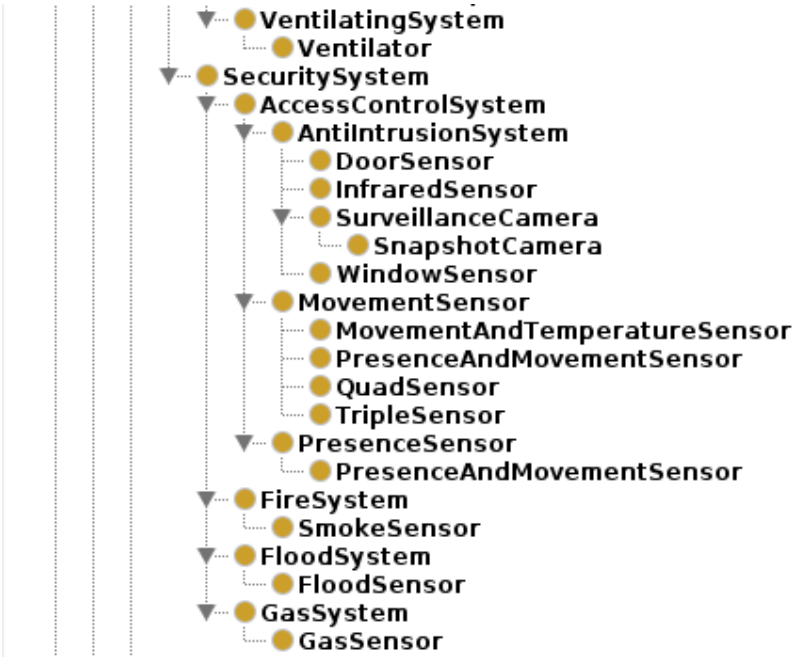
Di seguito c'è una lista di dipositivi rappresentabili in ShellML:


Controllable ≡ Device
Appliances
BrownGoods
Clock
CoffeeMaker
Communication
AnsweringMachine
Fax
Interphone
Phone
WirelessPhone
Computer
Entertainment
Dvd
HiFi
Radio
Tv
Fan
Printer
WhiteGoods
Boiler
Cooker
ElectricalCooker
GasCooker
Hotplate
DeepFreezer
Dishwasher
Fridge
Oven
ElectricalOven
MicrowaveOven
Stove
WashingMachine
HousePlants
ElectricalSystem
Acoustic
Buzzer
Siren
Actuator
AwningActuator
DoorActuator
SelfClosingDoorActuator
GateActuator
Pump
Shade
ShutterActuator
StepShutterActuator
ThermostaticRadiatorValve
WindowActuator


Control
Button
HeatingAndCoolingUnit
Knob
PumpController
RemoteControl
SceneSelector
ShadeController
ShutterButton
SingleScenarioControl
Switch
LevelControlSwitch
DimmerSwitch
ColorDimmerSwitch
OnOffSwitch
OnOffLightSwitch
RockerSwitch
Thermostat
ToggleRelay
Lighting
Lamp
DimmerLamp
DimmableLight
ColorDimmableLight
SimpleLamp
FlashingLamp
SimpleLamp
Meter
ElectricityMeter
CurrentMeter
SinglePhaseCurrentMeter
SinglePhaseElectricityMeter
ThreePhaseCurrentMeter
ThreePhaseElectricityMeter
EnergyMeter
MultiTariffEnergyMeter
MultiTariffSinglePhaseEnergyMeter
MultiTariffThreePhaseEnergyMeter
SingleTariffEnergyMeter
SinglePhaseActiveEnergyMeter
EnergyAndPowerMeteringLevelControllableOutput
SinglePhaseEnergyMeter
EnergyAndPowerMeter
EnergyMeteringPowerOutlet
SinglePhaseElectricityMeter
ThreePhaseElectricityMeter
ThreePhaseEnergyMeter
FrequencyMeter
SinglePhaseElectricityMeter
ThreePhaseElectricityMeter


SinglePhaseElectricityMeter
ThreePhaseElectricityMeter
PowerFactorMeter
MeteringPowerOutlet
SinglePhaseElectricityMeter
ThreePhaseElectricityMeter
PowerMeter
ActivePowerMeter
SinglePhaseActivePowerMeter
EnergyAndPowerMeter
EnergyAndPowerMeteringLevelControllableOutput
PowerMeteringLevelControllableOutput
PowerMeteringPowerOutlet
MeteringPowerOutlet
SinglePhaseElectricityMeter
ThreePhaseActivePowerMeter
ThreePhaseElectricityMeter
ApparentPowerMeter
SinglePhaseApparentPowerMeter
SinglePhaseElectricityMeter
ThreePhaseApparentPowerMeter
ThreePhaseElectricityMeter
ReactivePowerMeter
SinglePhaseReactivePowerMeter
SinglePhaseElectricityMeter
ThreePhaseReactivePowerMeter
ThreePhaseElectricityMeter
VoltageMeter
SinglePhaseVoltageMeter
SinglePhaseElectricityMeter
ThreePhaseVoltageMeter
ThreePhaseElectricityMeter
GasMeter
HeatingEnergyMeter
WaterConsumptionMeter
PowerDelivery
LevelControllableOutput
EnergyAndPowerMeteringLevelControllableOutput
PowerMeteringLevelControllableOutput
OnOffOutput
LampHolder
MainsPowerOutlet
EnergyMeteringPowerOutlet
MeteringPowerOutlet
PowerMeteringPowerOutlet
MeteringPowerOutlet
Sensor ≡ Sensor
Co2Sensor
DoorSensor
FloodSensor


FlowSensor
MultipleFlowSensor
SingleFlowSensor
GasSensor
HumiditySensor
QuadSensor
TemperatureAndHumiditySensor
InfraredSensor
LightSensor
QuadSensor
TripleSensor
MovementSensor
MovementAndTemperatureSensor
PresenceAndMovementSensor
QuadSensor
TripleSensor
MultiplePressureSensor
OccupancySensor
PresenceSensor
PresenceAndMovementSensor
PressureSensor
SmokeSensor
TemperatureSensor
TemperatureSensor
TemperatureSensor
WindowSensor
HVACSystem
AirConditioningSystem
CompositeThermostat
HeatingAndCoolingUnit
HeatingSystem
Heater
GasHeater
GasoilHeater
PelletHeater
MultipleFlowSensor
MultiplePressureSensor
PressureSensor
Pump
PumpController
SingleFlowSensor
TemperatureSensor
TemperatureSensor
MovementAndTemperatureSensor
QuadSensor
TemperatureAndHumiditySensor
ThermostatTemperatureSensor
TripleSensor
ThermostaticRadiatorValve
ThermostatTemperatureSensor




L'esempio seguente mostra la definizione di una lampada.
Essa ha nome "*SimpleLamp*" ed è descritta come una lampada che può solo essere accesa o spenta.
Possiede le funzionalità "*OnOffFuncionality*" e "*OnOffNotificationFuncionality*" e possiede i comandi "*OnCommand*" e "*OffCommand*" che prendono rispettivamente i valori "*on*" e "*off*".
Possiede uno stato di nome "*OnOffState*" che prende i valori "*on*" e "*off*".

Descizioni delle funzionalità "*SimpleLamp*":

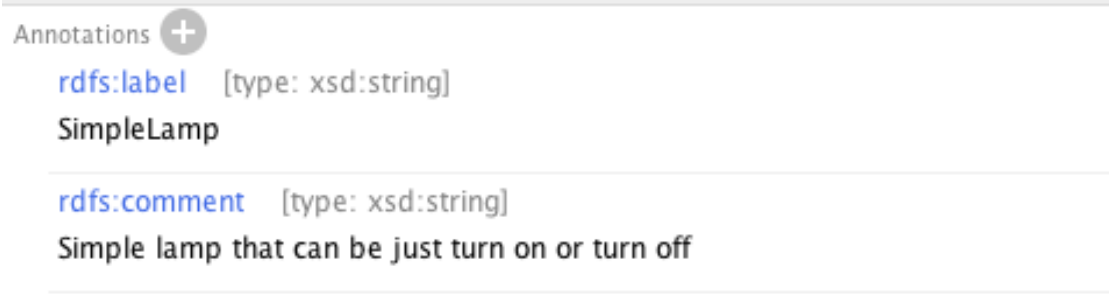



**Figura 2: descrizione della "*SimpleLamp*" in liguaggio naturale**

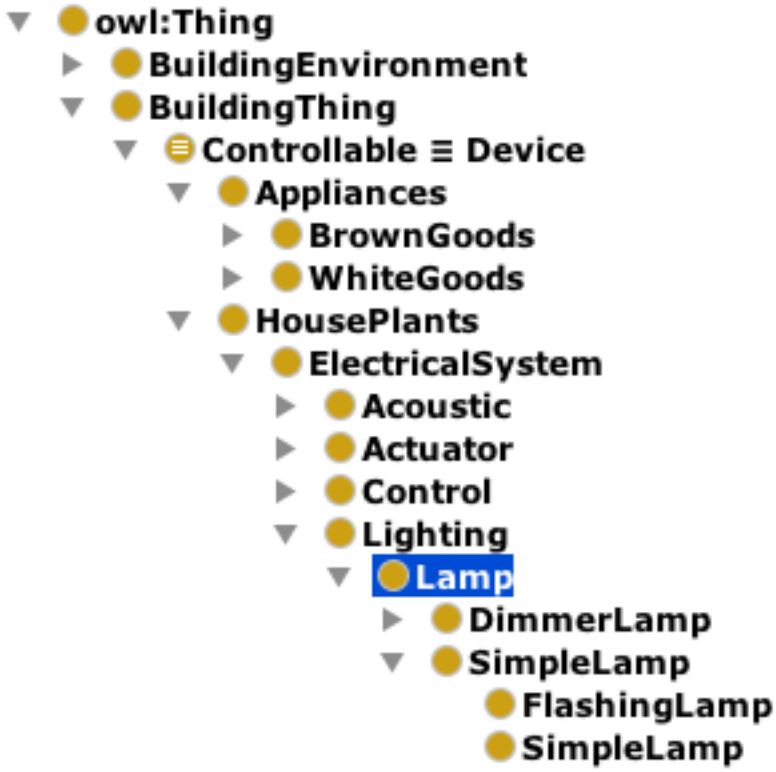



**Figura 3: posizionamento dell "*SimpleLamp*" all'interno dell'albero**


**Figura 4: definizione delle funzionalità della "SimpleLamp"**

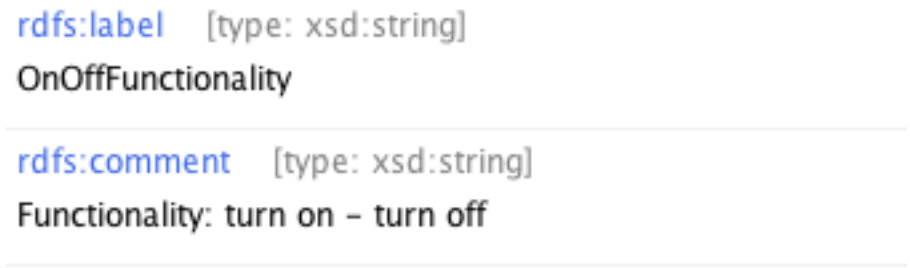



**Figura 5 : descrizione della "OnOffFunctionality" in linguaggio naturale**

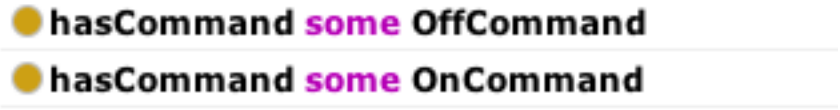



**Figura 6: tipologia dei possibili comandi per la "*SimpleLamp*"**


**Figura 7: contenuto del comando "*OffCommand*"**

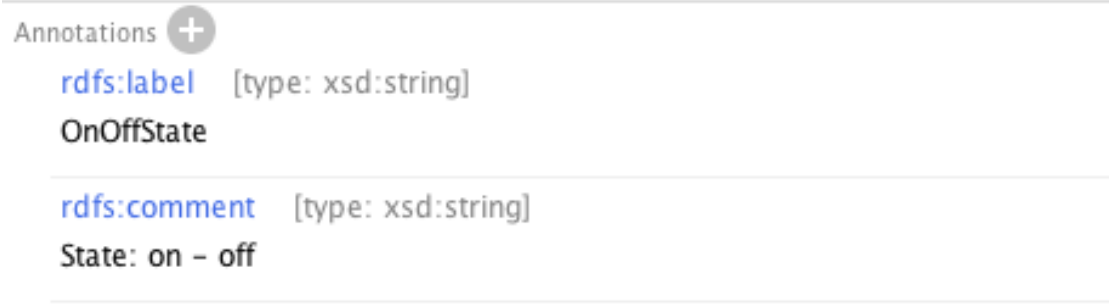



**Figura 8: descrizione del "*OnOffState*" per la "*SimpleLamp*", in linguaggio naturale**


**Figura 9: contenuto del "*OnOffState*"**

# 5. Ipotesi di comunicazione tra gli HOST

## 5.1. Configurazione delle associazioni tra i dispositivi

Definire le modalità di comunicazione tra gli HOST significa, in ultima analisi, descrivere come avvengono le interazioni tra i devices all'interno del sistema SHELL.
Occorre quindi stabilire le regole e le metodologie che indicano quali sono le associazioni tra i devices (chi parla con chi) e come avvengono. Esse operano conformemente alla topologia di rete del framework SHELL, in cui i nodi sono i devices (ospitati dagli HOST) con le loro funzionalità, mentre i rami sono le relazioni di associazione tra i device.
Tale tipologia si può rappresentare sotto forma di grafo, nel quale i nodi in grado di scambiarsi direttamente informazioni, sono collegati tra loro tramite uno o più rami.
Secondo tale rappresentazione, due nodi comunicano tra loro in modo logico, quindi astraendo dal livello fisico di comunicazione.
Si prevede che ogni istanza del modello del framework SHELL, abbia una diversa e sua propria configurazione di grafo e che questa venga formalizzata descrivendo ogni singola rappresentazione in formato XML.
Tale file XML (ASSOCIATION.XML), conterrà la descrizione di come si associano i dispositivi, quali sono i comandi che debbono essere messi in relazione tra loro e quali parametri dei comandi relativi alle varie funzionalità, dovranno essere presi in considerazione.

## 5.2. Creazione tabella delle corrispondenze

Creazione di una struttura dati in formato tabellare nella quale ogni entry si trova il mapping tra i comandi e il mapping tra i parametri, entrambi appartenenti alle rispettive funzionalità dei devices in gioco.
Tale tabella avrà come input le definizioni contenute nel file ASSOCIATION.XML. Essa verrà creata alla partenza del sistema SHELL e avrà contenuti diversi per ogni istanza del framework SHELL

## 5.3. Individuazione delle corrispondenze

Allo scatenarsi di un evento su un device X (es. interruttore) viene rilevato il comando da esso generato (es."*Off*") e ne viene cercata l'eventuale entry nella tabella. Tale entry conterrà la corrispondenza con il device Y (es. tra "*SimpleLamp*" e interruttore) e all'interno di questa, verrà scoperta l'associazione tra i rispettivi comandi e parametri dei comandi.

## 5.4. Trasmissione dei comadi

L'HOST in cui risiede il device X, invia al destinatario il corrispondente comando da eseguire sul device Y e quest'ultimo lo manda in esecuzione. Nel nostro esempio della "*SimpleLamp*" tale comando è anch'esso "*Off*", ma tale identità del nome corrispondente dei comandi non è vera in generale. Per esempio nell'associazione tra interruttore e DVD potremmo avere la corrispondenza tra i comandi "*Off*" dell'interruttore e "*standBy*" del DVD.

### 5.5. Funzionalità di gateway

Nel caso in cui uno o entrambi i device coinvolti fossero di tipo "wrapped", gli HOST implementeranno anche funzionalità di gateway, cioè sostanzialmente algoritmi di traduzione dei nomi delle funzionalità nel formato ShellML e viceversa. In questo caso i dispositivi in gioco apparterranno ad altri sistemi (altri framework/ bus HW e/o SW) e i gateway corrispondenti saranno quindi specifici per ogni tecnologia domotica da interfacciare.

### 5.6. Scoperta dei dispositivi esterni all'HOST

Allo start-up del sistema SHELL, i vari HOST comunicano in broadcast quali sono i dispositivi che essi gestiscono direttamente. I singoli HOST verificano nella propria tabella delle corrispondenze (ASSOCIATION.XML), se tra questi ci sono device (identificati dal DEVICE_ID) che essi debbono gestire, ma che sono esterni al proprio HOST. In caso affermativo viene associato ai DEVICE_ID esterni, l'indirizzo IP dell'HOST corrispondente. Tali indirizzi IP saranno poi utilizzati dalla rete per inviare i messaggi tra gli HOST, contenenti i comandi da eseguire.